

*AdvancedTCA*®

---

PICMG SFP.0

Revision 1.0

# System Fabric Plane Format Short Form Specification

**This short form specification is derived from the PICMG SFP.0 Revision 1.0 System Fabric Plane Format specification as approved on March 24, 2005 by the PICMG® Executive Membership. For guidelines on the design of the SFP.0 compliant boards and systems, refer to the full specification—do not use this short form for any design decisions.**



# SFP.0 – System Fabric Plane Format

---

## 1 INTRODUCTION

### 1.1 Purpose

This Short Form version of the SFP.0 specification is intended to provide an overview of the *System Fabric Plane (SFP)* Format. Note that most of the SFP.0 detailed low-level requirements have been removed in this Short Form version. Do not attempt to design to this Short Form specification.

### 1.2 Why SFP?

Many systems internally insert a protocol layer between Layer 2 (e.g. Ethernet) and Layer 3 (e.g. Internet Protocol). This is often referred to as a Layer 2½ protocol. Another common name for this type of protocol is a “Shim” (i.e. because it is inserted by the system between Layers 2 & 3).

The use of a Shim is very common on internal system fabrics. Many routers, media servers, firewalls, and other similar network equipment use these Shim protocol layers internally. But up to now these Shim layers have all been proprietary. In order to enable an ecosystem of Advanced TCA board vendors to interoperate with each other as a single system, a common standardized Shim layer needs to be defined. SFP defines this standardized Shim layer.

### 1.3 What is SFP?

The System Fabric Plane (SFP) framework is intended to be a standard way for modules to communicate in Comm Fabric type applications, such as:

- Routers
- Wireless Radio Network Controllers (RNCs)
- Voice over Packet Gateways
- Media Servers
- Firewalls
- Etc.

or any combination of the above. Note that these are all prime candidate applications for ATCA.

SFP is a layer that encapsulates many types of application protocols, such as:

- Internet Protocol Datagrams
- TDM/Voice (I-TDM)
- ATM (AAL-1, AAL-2, AAL-5)
- Wireless SDUs / PDUs (small packets for Wireless applications)
- Etc.

Note that SFP does not aim to be an end-user protocol. It exists only within the confines of a Converged Communications System (e.g. Content Processing Router).

## 1.4 What is a System Fabric?

A *System Fabric* is a Layer-2 network (e.g. LAN) that has additional protocols implemented in the endpoint nodes such that the collection of these nodes acts together as a single system. Note that a System Fabric may be implemented as:

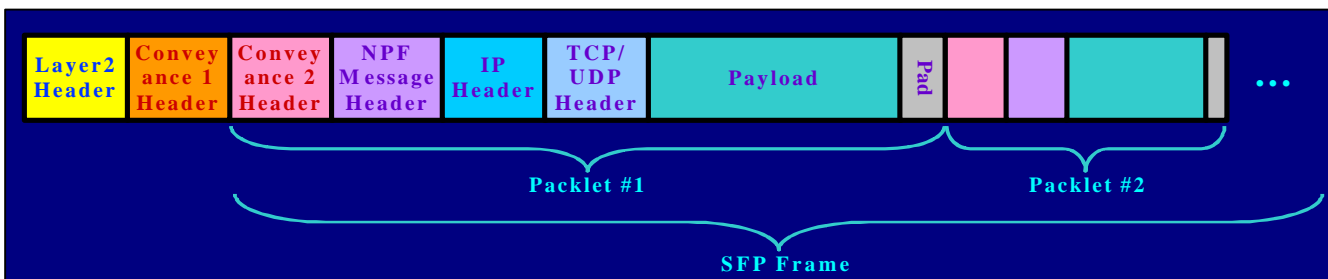
- a LAN between blades in a backplane
- a LAN that spans multiple backplanes
- a private LAN between Rack Mount Servers that form a system
- any combination of the above

## 1.5 SFP Encapsulations / Tunneling Examples

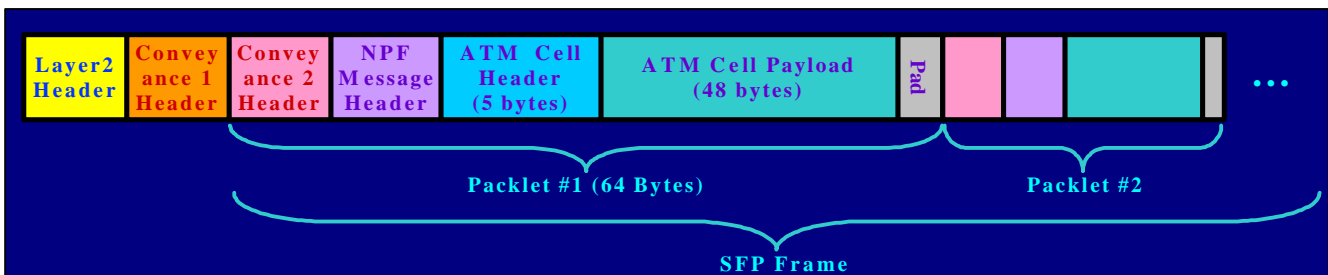
SFP can be used to encapsulate any other protocol. In other words, any other protocol can be tunneled through SFP. Some examples of these are given below. Note that SFP encapsulations are not limited to this set of examples.

Note that different encapsulations may be used within the same SFP Frame. For example, an IP packet, an ATM AAL-2 CPS packet, and an I-TDM payload may be multiplexed together as different packets of the same SFP Frame. SFP imposes no restrictions on what types of packet encapsulations may be multiplexed together.

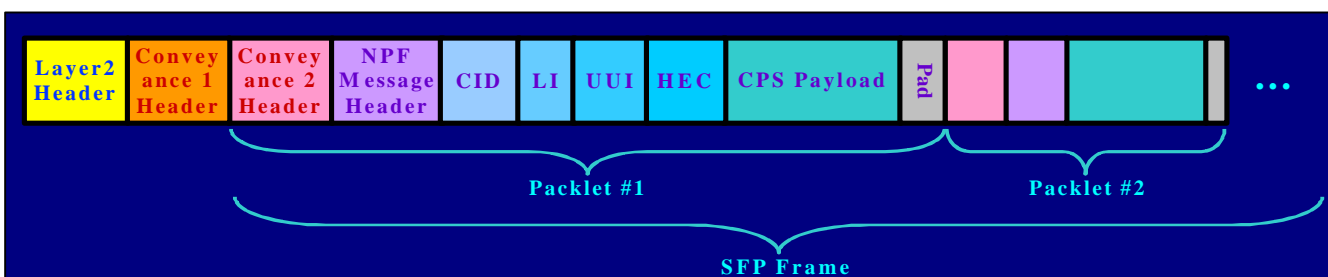
### 1.5.1 Internet Protocol Packet



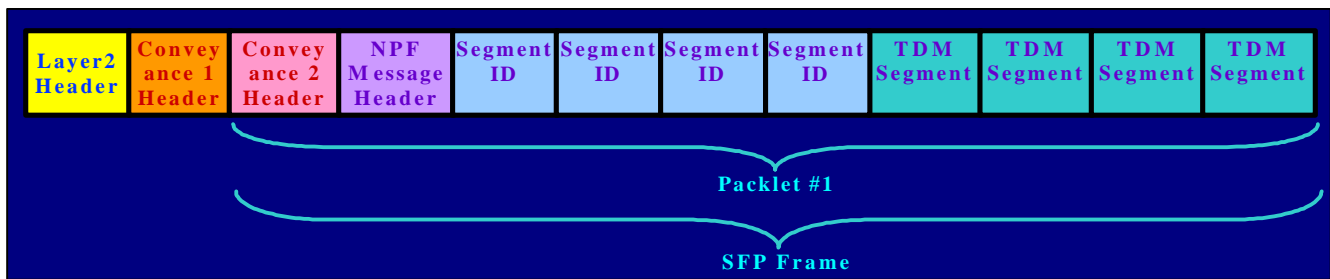
### 1.5.2 ATM Cell



### 1.5.3 ATM AAL-2 CPS packet



## 1.5.4 TDM/Voice (I-TDM 1ms Protocol)



The I-TDM protocol is specified in a separate document (SFP.1).

## 1.6 Scope

### 1.6.1 Layer2 Compatibility

SFP can ride on top of many different Layer2 Fabric protocols, such as:

- Ethernet
- Advanced Switching (AS)
- CSIX
- Infiniband
- Etc.

However, this specification will be limited in scope to Ethernet based Fabrics.

### 1.6.2 Flow Control

The SFP.0 headers do not currently specify any Flow Control or backpressure mechanisms. Such items are handled at Layer 2 (i.e. Ethernet layer).

### 1.6.3 Higher Protocol Layers

This document (SFP.0) does not specify all the protocols above it. Some examples of SFP encapsulations of existing protocols are given in section 1.5, but specifically the Internal TDM protocol which is layered on top of SFP.0 is specified in a separate SFP.1 document. Protocols for other future encapsulations may also be specified as additional SFP “dot specs” (e.g. SFP.2, SFP.3, etc.). All of these potential future protocols would presumably also be layered on top of SFP.0.

## 2 RATIONALE FOR SFP

Section 2 overviews the driving application level requirements that led to the creation of the SFP protocol. Note that the whole of Section 2 is for informational purposes only and does not contain any items for compliance testing.

The SFP headers are designed to:

1. Provide for rapid detection of errored or misrouted packets.
2. Provide for Segmentation / Reassembly of packets that are too large to fit within the MTU of the fabric.
3. Provide for Multiplexing of multiple small packets within a larger fabric MTU in order to reduce average Layer 2 overhead.
4. Provide a means to reduce or eliminate multiple redundant hash lookups.
5. Provide a means to carry ingress marking information.
6. Provide a means to extend the header.
7. Provide a modular interface to the low-level application using the fabric.
8. Be compliant with NPF Messaging.
9. Provide 64-bit alignment at header and payload boundaries to optimize operations by processors with 64-bit data paths.
10. Provide a means to transport a packet that is shorter than the Fabric's minimum frame size.
11. Provide an optional end-to-end system fabric integrity check.

Further elucidation of and descriptions of means to meet these criteria are provided in the following sections.

### 2.1 *Rapid Detection of Errored or Mis-Routed Packets*

#### 2.1.1 Errored Packets detectable by CRC

Most Layer 2 protocols provide a Layer 2 CRC that tells the receiver if the packet was modified during transit. If the modification was not intentional, it is called an *error*.

**Implication:** The fabric must be able to detect if a packet has been errored so that the control plane can take corrective action. It must also be able to do so as rapidly as possible.

Fabric switches typically drop errored packets. In this case, the intended receiver will never get the packet. To solve this particular problem, one introduces a flow label and a sequence number.

This flow label identifies the flow from the source node and its Virtual Output Queue (VoQ) to the destination node. Call this flow label a *Flow Bundle*.

Along with the Flow Bundle is a sequence number, called an *error detection sequence number*. This sequence number just increments and wraps. A single bit will suffice, since if a packet is mis-routed the next packet will be out of sequence and the control plane will be notified.

Thus, use of a Flow Bundle and an error detection sequence number of one bit is sufficient, in combination with a Layer 2 CRC, to rapidly detect errored packets.

However, it is possible that more than a few packets in a row are errored. The greater the number of bits in the error detection sequence number, the less likely that a multi-packet error could go undetected. Thus, more bits would be better, perhaps 4 to 8.

### 2.1.2 Misrouted Packets

Misrouting can occur without errors. For example, a software bug could write an erroneous entry into a switch chip's routing table. Then, an *unintended receiver* will start receiving the *misrouted flow*.

**Implication:** The fabric must be able to detect misrouted packets as rapidly as possible.

A Flow Bundle (as defined above) and *misroute detection sequence number* are sufficient to detect this condition. There are two cases:

1. If the unintended receiver does not have an active flow set up for the Flow Bundle, then the condition is detected immediately.
2. The unintended receiver does have an active flow set up for the Flow Bundle. Call this flow the *correct flow*. In this case, the two flows' sequence numbers will conflict, and the condition can be detected quickly.

To increase the probability of a mis-match between the two flows' misroute detection sequence numbers, the modulus of the misrouting detection sequence number should be somewhat higher than that of the error detection sequence number, e.g. 4 to 8 bits.

Thus, a Flow Bundle and a misroute detection sequence number are sufficient to detect misrouted packets.

## 2.2 Segmentation / Reassembly

The fabric must be able to carry application layer packets that are larger than the fabric MTU (maximum transmission unit). This is accomplished by segmenting the packet as it is placed onto the fabric and reassembling it when it is taken off the fabric.

**Implication:** The fabric must be able to perform Segmentation/Reassembly.

If there is completely reliable delivery, then a Flow Bundle (as defined above) plus an *EOP* (end of packet) bit is sufficient. The receiving algorithm is to always concatenate the current fabric packet onto the current application packet in memory. In addition, if the EOP bit is set, then send the current application packet to application, and start a new current application packet with nothing in it.

Individual fabric packets can be errored or not arrive, as described above, and this is detectable by the errors detection algorithm stated above. In order to recover rapidly, an *SOP* (start of packet) bit is also used. Use of an SOP bit provides every possible description of a fabric packet: start of packet, middle of packet (neither EOP nor SOP set), end of packet, and complete packet (EOP and SOP set).

The recovery algorithm is that after an errored or missing fabric packet is detected, the current application packet is reset, and all fabric packets are discarded until one with an SOP bit is detected.

However, if the errored or misrouted packet detection mechanisms fail, then the Segmentation/Reassembly algorithm can fail. This failure would appear as an SOP packet, followed by a number of middle packets (neither EOP nor SOP), followed by an EOP packet. Some of the middle packets could be dropped, or worse, both an EOP and an SOP packet along with the middle packets. The result would be packet with data for one destination being sent to another destination.

To avoid this failure scenario, the errored and misrouted packet detection mechanisms must be as robust as possible, implying that more rather than fewer bits are needed.

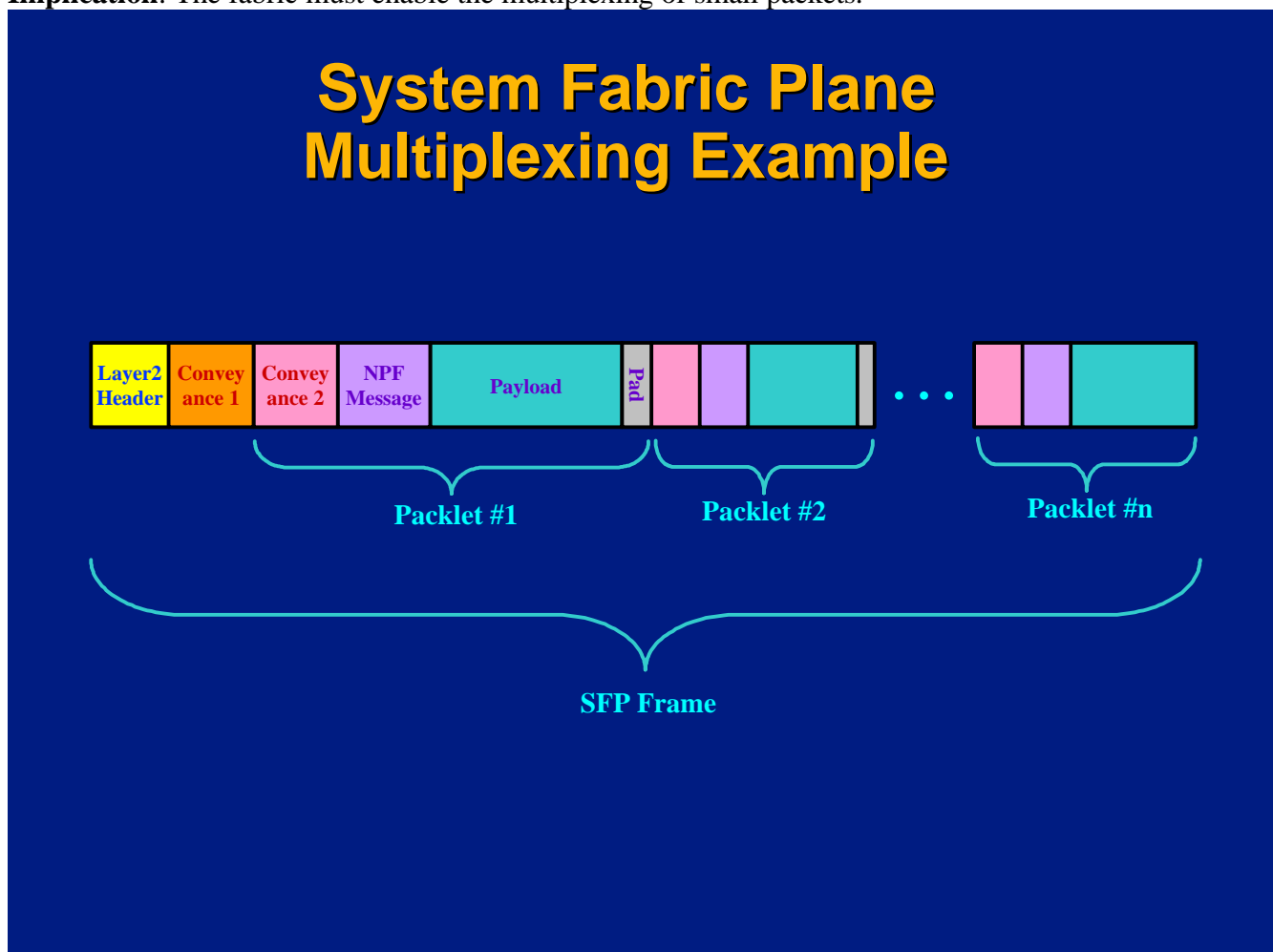
In essence, one can define that a Segmentation/Reassembly sequence number is required, again based upon the Flow Bundle, on the order of at least 8 bits.

However, SOP is not absolutely required. With just an EOP and Segmentation/Reassembly sequence #, errors will be dealt with adequately. The only issue is that you perhaps lose 2 application packets when a packet is errored. However, if we are running tight on bits, it may be better to have an extra Segmentation/Reassembly sequence # bit than an SOP bit. Bottom line: SOP is nice to have but not required.

## 2.3 Multiplexing

Multiplexing is needed in order to reduce the overhead associated with sending many small packets over the fabric.

**Implication:** The fabric must enable the multiplexing of small packets.



Section 4.1 includes a tradeoff analysis that shows the potential advantages of multiplexing.

A key software element used for SFP Multiplexing is the Virtual Output Queue (VoQ). This is an output queue for all traffic that goes to a specific destination on the fabric. VoQs are used in Comm Fabrics to eliminate congestion in fabric switches, so VoQs are not unique to SFP multiplexing.

For SFP Multiplexing, packets, or parts of packets, all on the same VoQ, are each encapsulated with an SFP header, to create *packlets*, which are concatenated into an SFP Frame, which is sent to the destination over the fabric.

At the destination, the SFP Frame is De-Multiplexed. Each packet within the SFP Frame is extracted and placed in the appropriate destination queue.

The only fields required for multiplexing are a length field and a last packet field. Note that a Length field is also required for Short SFP Frames (section 2.10), so the only field that is unique to Multiplexing is the last packet field.

### 2.3.1 Last Packet Field

When multiple packets are put together in an SFP Frame, the parser needs to know when the last packet has been reached. In SFP Frames longer than the minimum Layer2 frame-size, the last packet could be algorithmically determined by looking at the length of the Layer2 frame and the lengths of the individual packets. However, when the SFP Frame is shorter than the minimum Layer2 frame-size, there must be some way to determine the last packet without relying on the Layer2 frame-size. A field defining that a particular packet is the last one in the SFP Frame satisfies this requirement.

The last packet fields need only be a single bit.

Once available, an additional use for the last packet field is to simplify the packet parsing code. Given the last packet bit, the parsing code does not have to perform the ongoing length computation described above in order to determine the last packet. Thus, the last packet field provides a small computational benefit, as well.

### 2.3.2 Length

Each packet must have a length defined. If there were only one packet in an SFP Frame, then the length could be derived from Layer 2. However, when there are multiple packets in an SFP Frame (which is the multiplexing case), one must know how long the individual packets are.

The Length field also solves the Short SFP Frame requirement (see section 2.10).

## 2.4 Eliminating Redundant Hash Lookups

When a packet first comes into a system from outside, it must be classified. This is typically done by a hash lookup, (e.g. on an IP packet's headers).

If the results of this lookup are not saved somehow, then every intermediate node that the packet traverses will have to perform the same lookup.

That information is saved in an application Flow ID, which is attached to the packet and used whenever it enters a node to provide rapid classification and indexing into the receiving application's data structures.

## 2.5 Carrying Ingress Marking Information

When a packet arrives from outside the system, and after it is classified and metered, a marking operation is applied if appropriate. The marking performs two functions: it defines the service level that the packet should receive; and it determines whether the packet is within its service level agreement (*SLA*).

The packet will be determined to require a certain *service level*, which then must accompany the packet, as it determines in which specific Virtual Output Queue (VoQ) for a given destination node the packet will be placed.

For example, VoIP packets will have a real-time service level; TCP packets destined for control elements in the system will have high priority service level because they are most likely control traffic; TCP packets destined for routing through the system or to be terminated in deep packet inspection node will have only best effort service level applied, unless they are part of a specific SLA. The service level determines which VoQ the packet is placed in when it is forwarded across the fabric. The real-time VoQ is serviced at a higher priority than the best effort VoQ.

The service level is most likely encoded in the application Flow ID.

If the packet is in a flow for which there is an SLA, then it will be metered and marked. For example, a possible implementation is that if the packet is:

1. conforming to the SLA then it will be marked “green”
2. moderately out of conformance then it will be marked “yellow”
3. completely out of conformance, it will be marked “red”

When congestion occurs as packets flow through the system, those marked “red” will be preferentially dropped, then “yellow”, and “green” only as a last resort.

By convention, the marking stating whether the packet is within or out of its SLA is called *color*.

## 2.6 Extending the Header

Encapsulation protocols generally provide a field that defines what comes next. This field provides information to the decapsulation process that allows it to decide how to further parse the packet. For example, Ethernet has an Ethertype; InfiniBand has a Next Link Type.

This field is not strictly necessary. The destination may well know exactly what is in the packet without needing any further information about its type.

But if this field is necessary, then in order for the SFP header to be compliant with NPF, this field must be one byte in length, the first byte of the NPF Messaging header, and called SLOT\_CFG\_ID.

## 2.7 Modular Low-Level Application Interface

The SFP must present a clean interface to the low-level applications that use it. For example, low-level applications may be in Firmware, Drivers or Network Processor MicroEngines.

Given low-level coding issues, modularity will be relative. However, at the very least, there must be a clean separation between data elements that are used by the application and those that are used by the fabric.

For example, it seems unacceptable to expose the concept of a purely fabric specific data element such as an error detection sequence number to the low-level application.

## 2.8 NPF Messaging

The Network Processor Forum has defined a set of messaging fields and protocols for setting up these fields. Note that this NPF messaging is not specific to Network Processors. It may be used by any processor technology.

However, NPF messaging is optimized for performance. For example, NPF messaging does not use traditional TLV (Type, Length, Value) type fields as this would significantly decrease performance.

Since Comm Fabrics are also typically optimized for performance, SFP has adopted a specific implementation of NPF messaging. This NPF messaging layer is one of the two layers of SFP. The other SFP layer is called the “Conveyance Layer”. These 2 SFP layers will be detailed later in this document.

## **2.9 64-bit Alignment of Headers and Payloads**

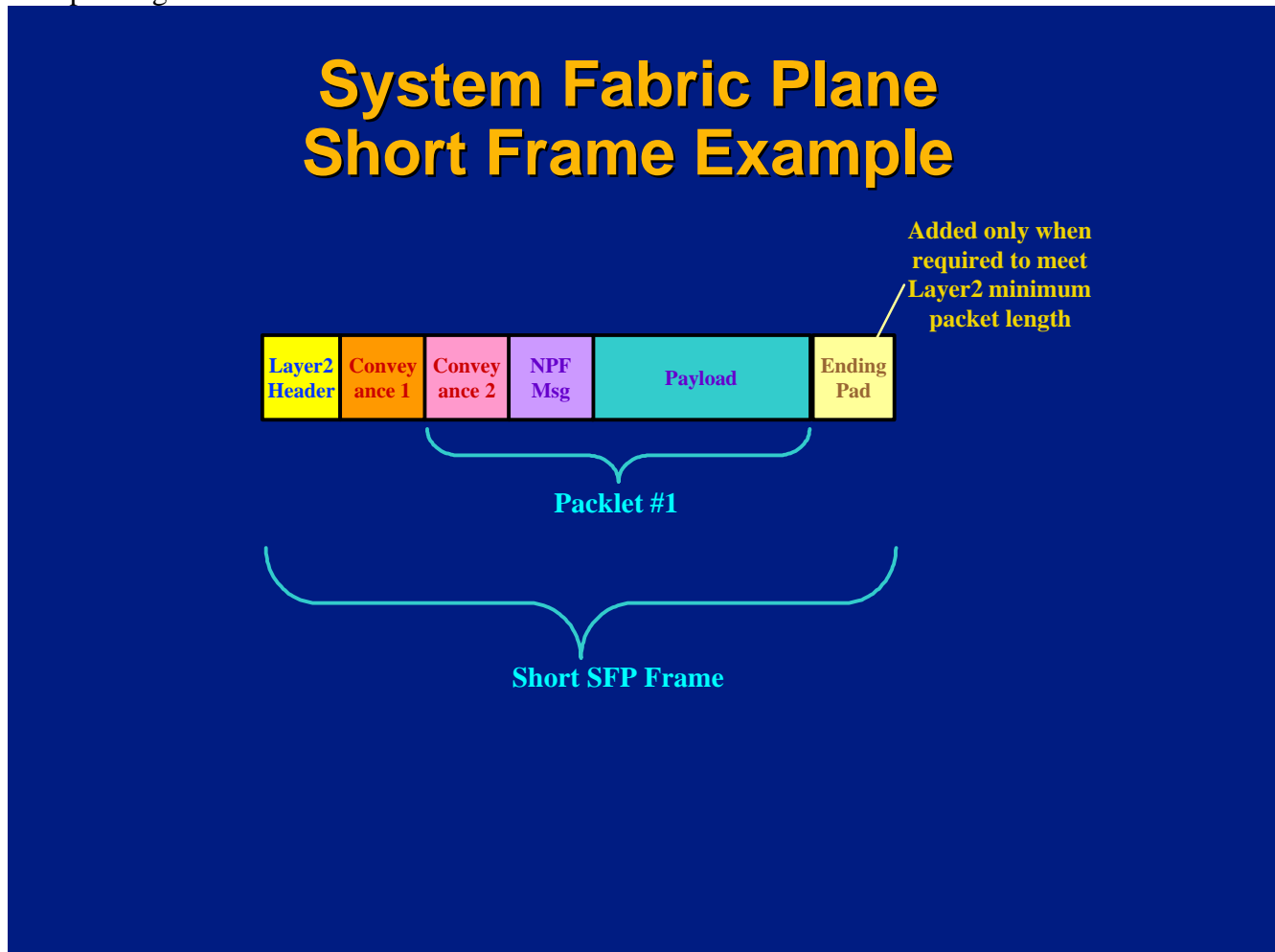
For Multiplexing, Segmentation and ITDM processing, aligning the fields on 64-bit boundaries is critical to performance when using processors with 64-bit data paths. Therefore, the SFP headers in general must be a multiple of 64-bits.

The vast majority of today’s processors have 64-bit data paths (e.g. Pentium® Processor, Sparc®, PowerPC® Network Processors, etc), so this is not an Intel-specific thing. Specifically for ITDM, starting on a 64-bit boundary is a big performance advantage on any processor with a 64-bit data path. This would include a server on a private LAN.

## 2.10 Short SFP Frames

If a small packet (e.g. AAL-2 CPS packet) is placed by itself into an SFP Frame, it might be that the SFP Frame will not fill a minimum Layer 2 packet. In this case, there must be padding at the end of the SFP Frame. There must be a way to distinguish this padding from the real payload.

A Length field in the SFP headers allows the receiver to determine the real length and discard any left over padding.



## 2.11 End-to-End System Fabric Integrity Check

Although the layer 2 protocol of the fabric typically performs a hop by hop integrity check (e.g. CRC, vertical/horizontal parity, etc), there is typically no layer2 integrity checking that spans the fabric system fabric endpoints. The SFP checksum implements such an end-to-end integrity check. The use of the checksum is optional.

## 2.12 Terminology

- A **System Fabric** is a Layer-2 network (e.g. LAN) that has additional protocols implemented in the endpoint nodes such that the collection of these nodes acts together as a system. Note that a System Fabric may be implemented as:
  - a LAN within a backplane
  - a LAN that spans multiple backplanes
  - a private LAN between Rack Mount Servers
  - any combination of the above
- An **application packet** is the chunk of data that an application wishes to send over the fabric. It is usually accompanied by some metadata.
- A **packet**, when used herein without other qualification, is an application level packet.
- **Packet metadata** includes, but is not limited to, the length of the packet and an application flow label. It likely includes possibly a packet type (e.g. control or data).
- An **application Flow ID** is a tag that makes sense to the application.
- A **packlet** results from encapsulating either an entire packet, or a fragment of a packet, within an SFP header. Multiple packlets are combined into an **SFP Frame**, to which is attached a Layer 2 header. The SFP Frame is sent over the Layer 2 Fabric.
- A **Flow Bundle** identifies a group of application level Flow IDs that share the same logical path over the fabric. Specifically, the **Flow Bundle** identifies the:
  - flow from a given source node at the destination node
  - priority or “class of service” over the fabric
  - geographical path over the fabric. For example, if there are 2 Ethernet LANs in a given Fabric (i.e. for redundancy), and both LANs are active (i.e. active-active redundancy scheme), then a packet may flow from a given source node to a given destination node over either LAN, which represents 2 active geographical paths.

## 2.13 Architecture

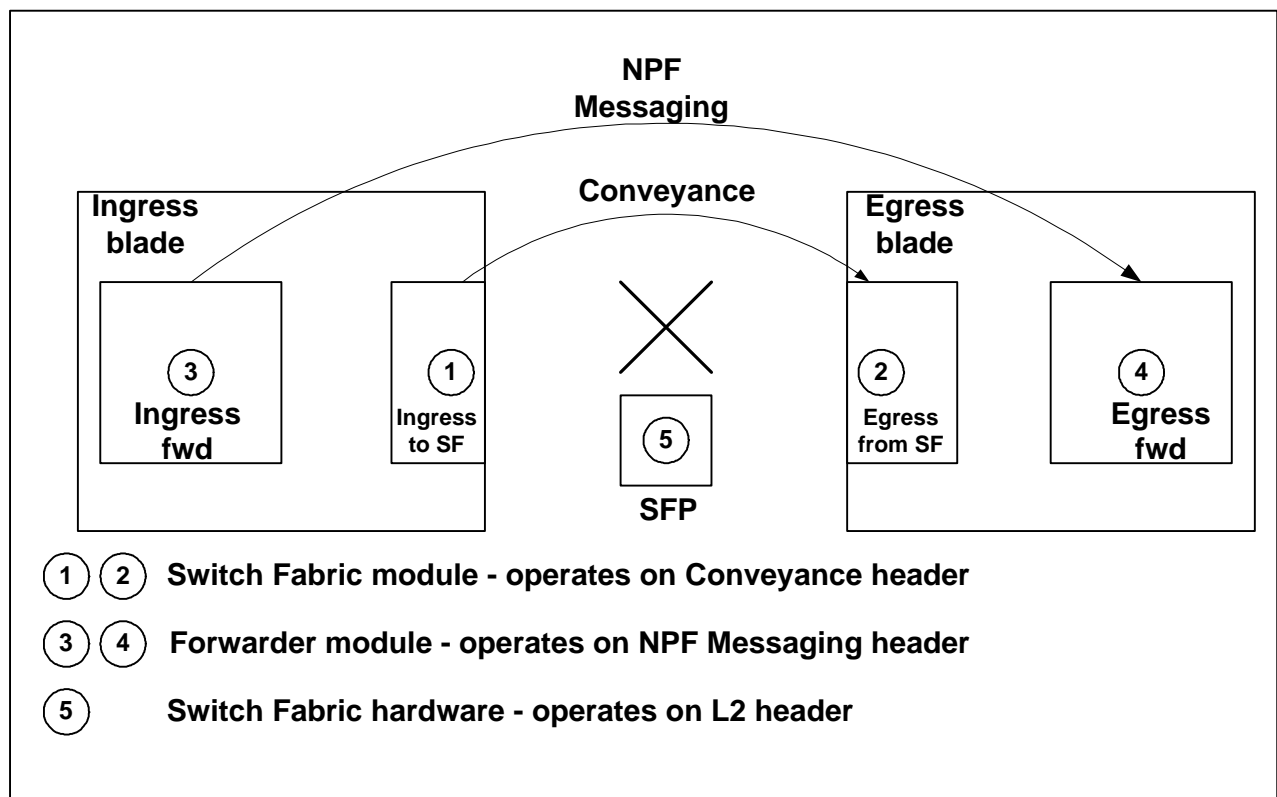
### 2.13.1 Principles

The proposed architecture of the solution follows these principles:

1. Header layer independence:
  - a. Conveyance header is for system fabric related info
  - b. NPF Messaging header is for application related info
  - c. Conveyance header does not depend on Messaging header, and vice versa
2. Flow Label meaning is negotiated by application (signaling)

### 2.13.2 Architecture

The architecture of the solution is based upon the principle of layering. In the picture below, flows between entities 3 and 4 (applications) are carried over the transport provided by entities 1 and 2 (the fabric ingress and egress entities). The layer controlled and used by entities 1 and 2 is the Conveyance layer. The layer controlled and used by entities 3 and 4 is the NPF Messaging layer.



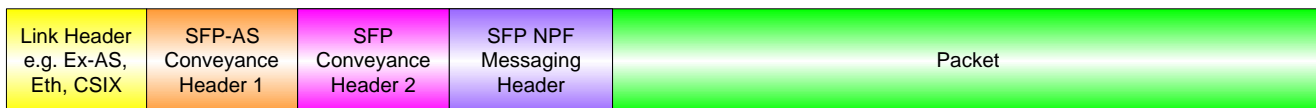
### 3 SFP PROTOCOL

This section defines the SFP protocol.

The various criteria described in Section 2, for the most part, carry directly down to fields that are required in the SFP header. For example, for multiplexing we need a packet length, and a last packet bit. For carrying service level and marking information we need appropriate fields, etc.

This Short Form version of the SFP.0 specification is intended to provide an overview of the *System Fabric Plane (SFP)* Format. Note that most of the SFP.0 detailed low-level requirements have been removed in this Short Form version. Do not attempt to design to this Short Form specification.

#### 3.1 SFP Specific Fields



SFP includes the following 3 headers:

- **SFP Conveyance Header 1.** This occurs once per SFP Frame.
- **SFP Conveyance Header 2.** This repeats for every Packlet within the SFP Frame.
- **SFP NPF Message Header.** This repeats for every Packlet within the SFP Frame. However, if Segmentation/Reassembly is required, the NPF Message Header is only valid on the 1<sup>st</sup> segment (Start of Packet). See section 3.5 for example of this.

The contents of each header are implemented as shown in the table and figure below:

<b>SFP Conveyance Header 1</b>	6-10 bytes
MPLS Label (contains Flow Bundle)	32 bits
Timestamp (for time-based applications or for measuring fabric latency)	16 bits
Pad (if required to align SFP Conveyance Header 2 on 64-bit boundary)	0-32 bits
<b>SFP Conveyance Header 2</b>	5 bytes
Sequence Number (increments within Flow Bundle context)	8 bits
SOP/EOP (for Segmentation and Reassembly)	2 bits
Last Packlet (for Multiplexing)	1 bit
Reserved (for Future Use)	2 bits
Length of Packlet (to 2KB)	11 bits
Checksum (optional use)	16 bits
<b>SFP NPF Messaging Header</b>	3+N bytes
UID (Application/Instance/Format Flow Label)	24 bits
Optional NPF Messaging Fields (e.g. Color, Timestamp, etc)	N bits

Refer to the full specification of SFP.0 for specific format variations and bit locations.

## 3.2 SFP Conveyance Header 1

This header occurs once per SFP Frame.

Refer to the full specification of SFP.0 for detailed requirements on this header.

### 3.2.1 MPLS Label (32 bits)

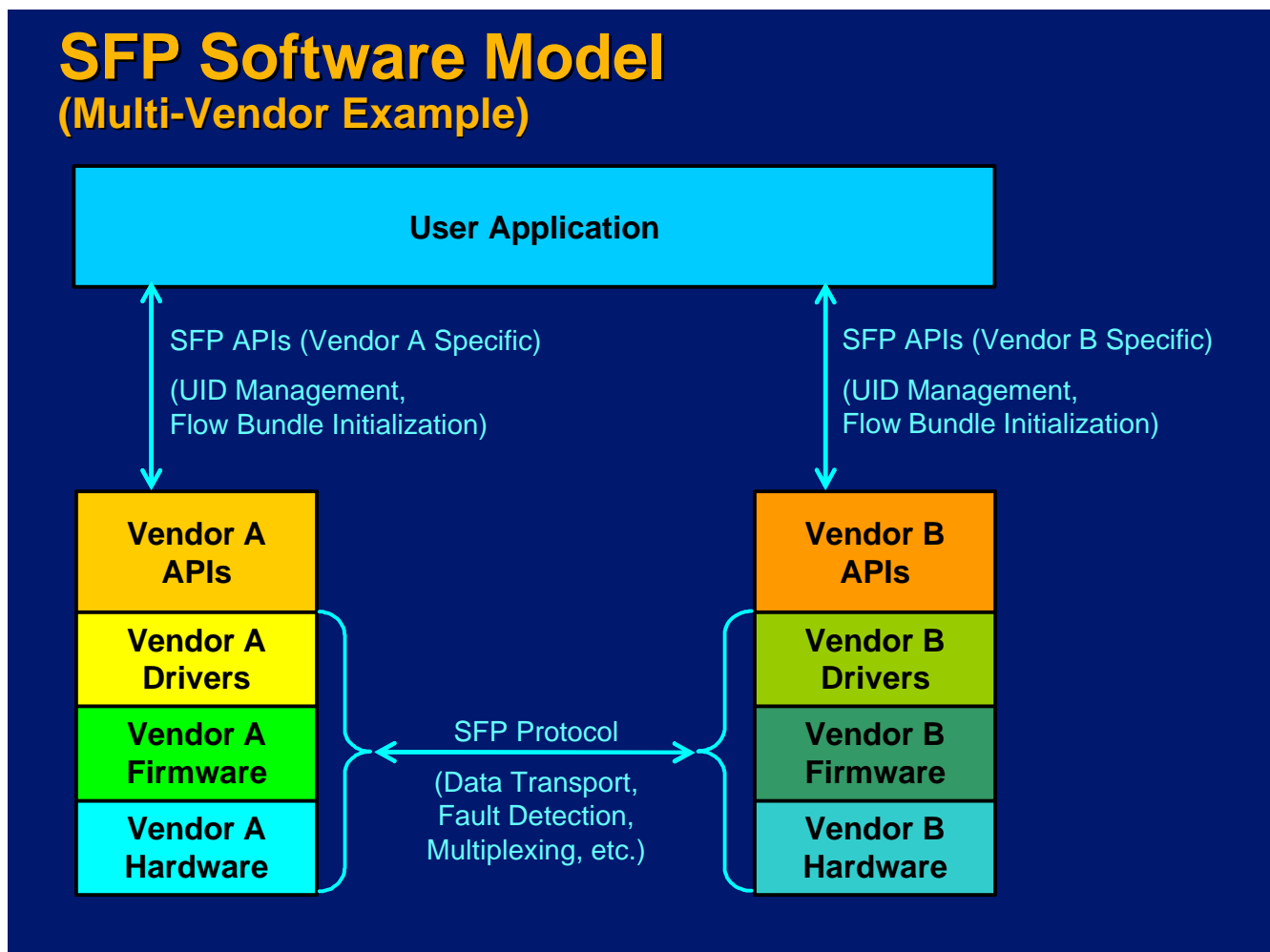
In the case of SFP, the MPLS Label represents the Flow Bundle. The Flow Bundle identifies:

1. The Source Node on the Fabric that produced this SFP Frame.
2. The class-of-service/priority on the Fabric
3. The path of the connection over the Fabric (if multiple active paths exist between 2 endpoints).

Note that, for many implementations, these 3 parameters essentially correspond to the Source Virtual Output Queue that originated the SFP frame.

The path portion of the Flow Bundle generally corresponds only to active paths, and not to inactive standby paths for redundancy. In particular, for active/standby redundant fabrics, it is generally very useful to have the same Flow Bundle assignments persist after a failover. Therefore, the failover can happen much more quickly. See section 4.3 for details.

In order to insure an interoperable ecosystem of different vendors SFP implementations, the following software model is used.



The User Application initializes all the Flow Bundles (i.e. 20-bit Labels within the MPLS Label field) for all SFP endpoints via the Vendor Specific APIs.

Different System Fabrics can allocate the Flow Bundle number space to each of the 3 parameters (i.e. source, class, and path) differently, depending on specific requirements of the system. Also note that 20 bits is much more than typically required to hold the 3 parameters. For example a multi-chassis system that has 256 blades, 8 priorities and 4 paths over the fabric would use only up to 8192 Flow Bundles, and this would be a large system.

The User Application should allocate the Flow Bundle number space efficiently so as to minimize the highest Flow Bundle number.

In addition, the Vendor Specific APIs may impose restrictions on the maximum Flow Bundle number in order to ease the implementation of the destination Flow Bundle look-up tables.

Note that the requirements above allow the Flow Bundle to be a destination resource. In other words, the destination may have a fixed array of state machines to track sequence numbers, segmentation / reassembly state, etc. for each Flow Bundle. In the case that the total number of Flow Bundles in a system exceeds the number of Flow Bundles supported by a particular destination implementation, that destination implementation can still be used with limitations. In other words, large systems may be built where not every source can talk with every destination over every path at every priority. In this case, the Flow Bundle values going to 2 different destinations may be the same. So for large systems, the Flow Bundle / MPLS Label may not uniquely identify a source/destination pair.

For qualifying the source node portion of the Flow Bundle number space, the only requirement is that no two source nodes talk to the same destination using the same Flow Bundle.

So for small systems, a normal MPLS Label distribution scheme should suffice. In a large system, other Flow Bundle / MPLS Label allocation algorithms may be necessary.

### 3.2.2 Timestamp (16 bits)

This field may be used for time based applications (e.g. I-TDM) in order to detect dropped, duplicated, or out-of-order packets (e.g. for fault recovery scenarios).

In addition, this field may be used to determine the amount of time it takes for a given SFP frame to traverse the SFP Fabric (i.e. delivery latency). The SFP delivery latency may then be used to identify paths of queuing congestion in the fabric.

The source node encodes the Timestamp field using 5 microsecond units. For example, if the Timer value increases by one, this means that 5us of time has elapsed. If the Timer value increases by 200, 1 millisecond of time has elapsed.

There are no SFP.0 requirements on the accuracy of the timer. Also, any use of the Timer field at the destination is optional and is not specified here.

Note that even though the Timer uses 5us units, there are no requirements to have a local time-base which increments by 5us or less. For example, if a given implementation only has a 1 millisecond local time-base available, then the source node simply increments the Timer field by 200 for each 1ms increment of the local time-base. However, if there is a desire to measure the delivery jitter/latency of certain packet flows across the fabric, then a higher resolution time-base would be beneficial.

### 3.2.3 Pad (0-32 bits)

This field is sized such that the SFP Conveyance Header 2 starts on a 64-bit boundary. If the Layer 2 header + MPLS Label + Time Stamp is already a multiple of 64-bits, the SFP Pad field is not used.

### 3.3 SFP Conveyance Header 2

This header repeats once for every packet in the SFP Frame.

Refer to the full specification of SFP.0 for detailed requirements on this header.

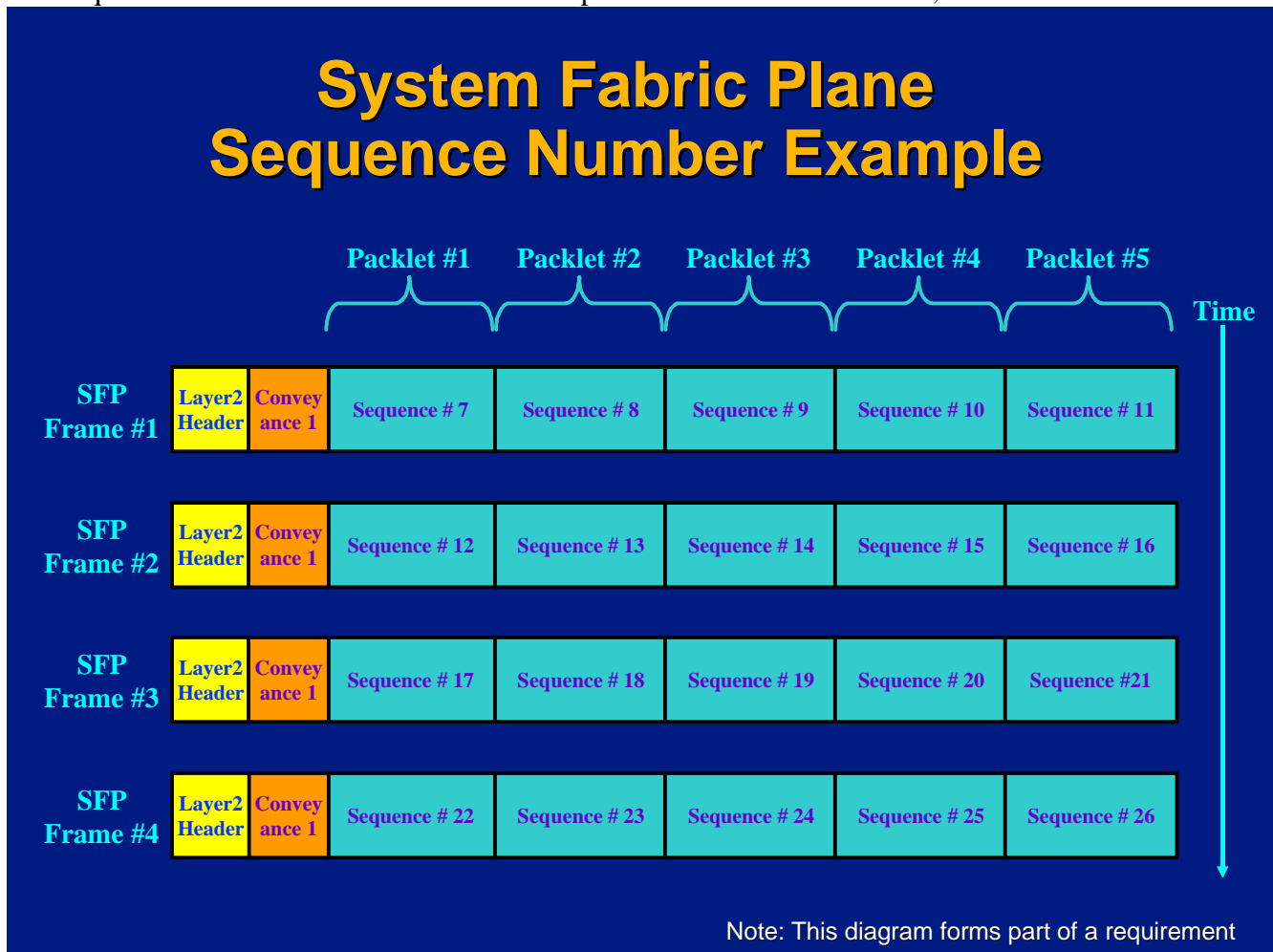
#### 3.3.1 Sequence Number (8 bits)

The SFP sequence number is used for multiple purposes, as listed below:

- Error detection
- Misroute detection
- Segmentation and Reassembly
- Multiplexing

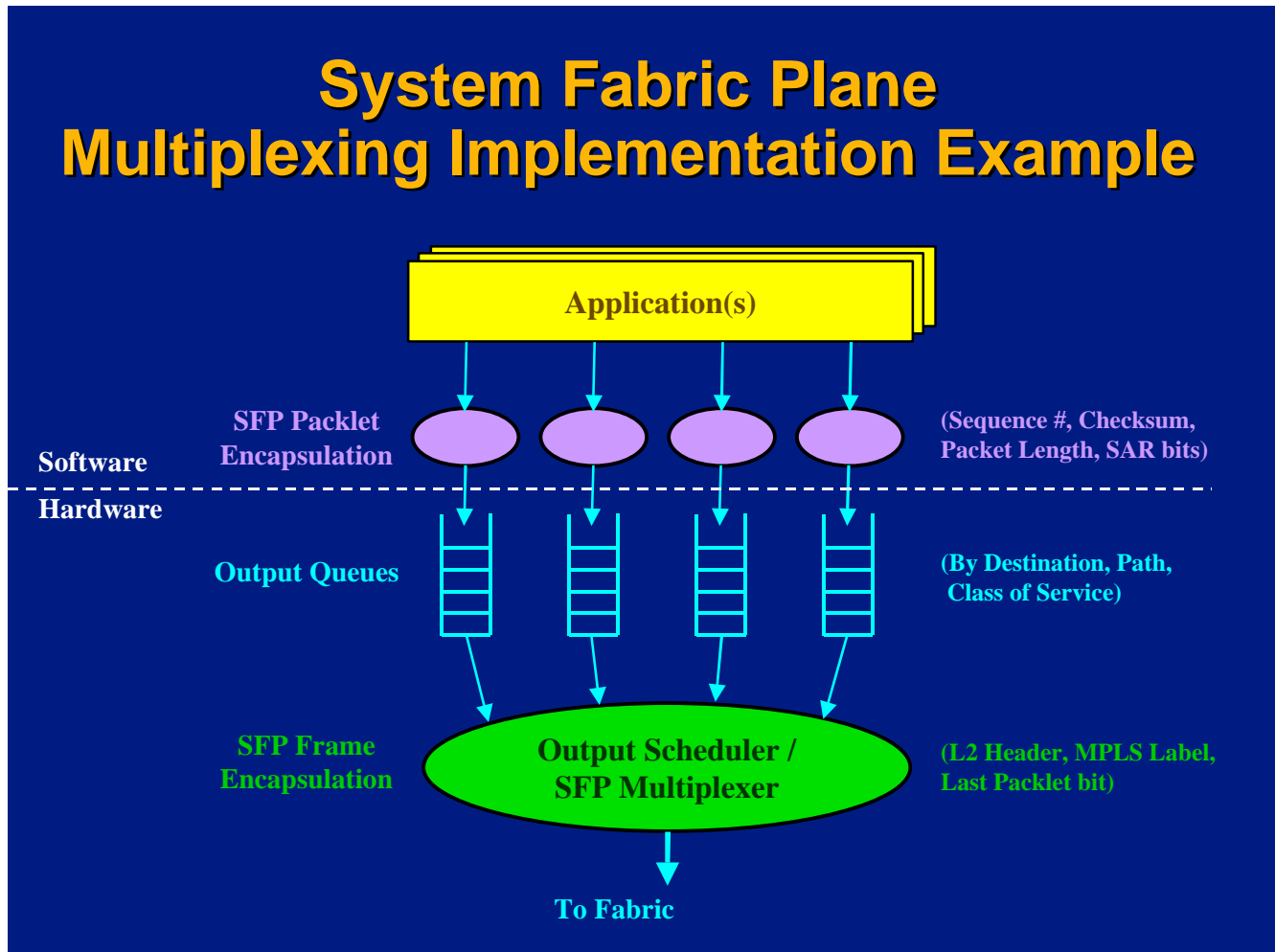
The sequence number increments within the context of a Flow Bundle (see section 3.2.1).

The sequence number also increments for each packet within an SFP Frame, as shown below.



As packets are de-multiplexed from an SFP Frame, a sequence number in the Conveyance Header of each packet within the SFP Frame provides a redundant check that the de-multiplexing parser is in the correct state and position in the SFP Frame.

Also, having a sequence number in each packetlet allows different implementations of SFP to append the sequence number at different layers. For example, one implementation may be that the SFP Conveyance layer Encapsulation and Segmentation is implemented in software, but the VOQ and SFP multiplexing functions are performed by hardware. In this scenario, sequence numbers are added in software while hardware performs the multiplexing.



This is not the only recommended implementation of SFP. It's just that the SFP sequence number method shown above is flexible enough to accommodate this type of implementation, as well as other implementations. Section 3.5 also touches on this flexibility.

Note that an additional sequence number may be required for certain types of application packets (e.g. reliable connection for control plane data). This additional sequence number may go in the NPF header or in various application layer headers (e.g. TCP). These additional sequence numbers are unrelated to the conveyance layer sequence number.

### 3.3.2 SOP/EOP

These bits are used to perform segmentation / reassembly of application packets that are too big to fit in a single fabrics' MTU.

### 3.3.3 Last Packetlet

This bit indicates whether or not this is the last packetlet of the SFP frame (see section 2.3 for an overview on SFP multiplexing).

### 3.3.4 Reserved

These 2 bits are reserved for future use.

### 3.3.5 Length of Packlet

This field is used for Multiplexing and Short Frames (see sections 2.3 & 2.10).

### 3.3.6 Checksum

The Checksum is used for an optional End-to-End System Fabric Integrity Check (see section 2.11).

The use of the SFP Checksum is optional. Destination nodes may, or may not, verify the Checksum. Source nodes may, or may not, calculate the checksum.

## 3.4 SFP NPF Messaging Header

This header repeats for every Packlet within the SFP Frame. However, if Segmentation/Reassembly is required, the NPF Message Header is only valid on the 1<sup>st</sup> segment (Start of Packet). See section 3.5 for an example of this. Note that the SFP NPF Messaging Header is a specific base implementation of the proposed NPF Messaging Header.

Refer to the full specification of SFP.0 for detailed requirements on this header.

### 3.4.1 UID (Application/Instance/Format Flow Label)

The 24-bit UID field identifies:

- The Application
- The Instance within the Application
- The Format of the message

The allocation of the UID number space is done so that specific processes at the destination node can do a table look-up fairly easily. For this reason, the destination node is responsible for the allocation of the UID number space. This allocation may be implemented in a number of ways, such as:

- The Vendor Specific API dynamically allocates UIDs on request.
- The Vendor simply documents the UID number space restrictions, and the User application implements these restrictions.
- Etc.

Refer to section 3.2.1 for details on the Software Model.

### 3.4.2 Additional Optional NPF Messaging Fields

For certain types of packets, additional fields may be added to the NPF header (e.g. Color, Ingress Timestamp, etc).

Note that certain types of encapsulation protocols (e.g. I-TDM) require the payload to start on a 64-bit boundary. For these types of payloads, it is desirable not to add additional NPF messaging fields in order to maintain 64-bit alignment with minimal header bytes.

For other types of encapsulated protocols that do not require 64-bit alignment (e.g. IP Datagrams), additional fields may be added to the NPF header as needed.

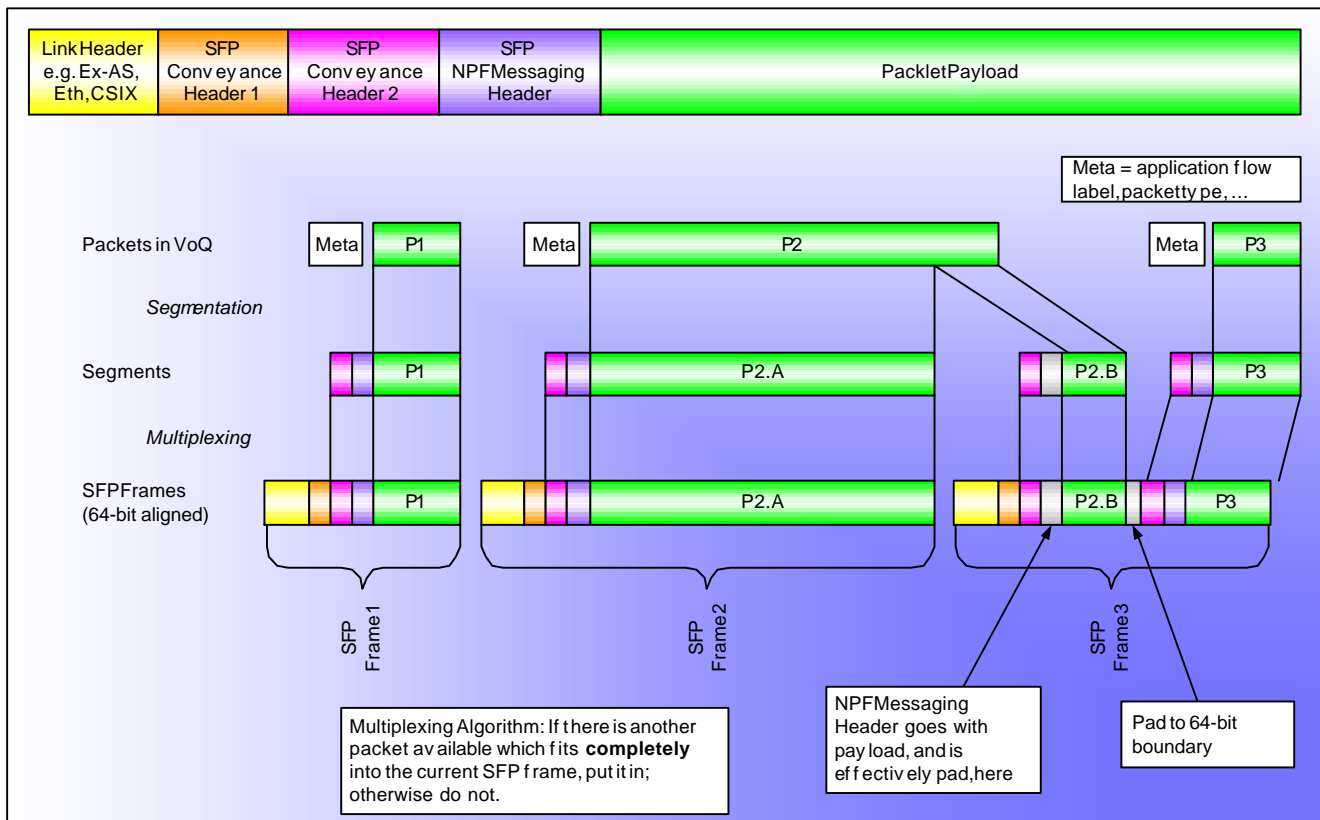
### 3.5 SFP Segmentation/Multiplexing Details

All SFP destination endpoints must be capable of demultiplexing multiple packets in a single SFP frame.

The algorithm for multiplexing is as follows: If there is another user packet for the same Flow Bundle that fits completely into the current SFP frame, then put it in, otherwise do not. The source node will NEVER segment a user packet that would normally fit into a single SFP Frame.

An example of SFP is shown below, working on 3 packets in a VoQ:

1. The first packet in the VoQ is converted to a packet. Because the next packet is large, it will not fit completely into the rest of the SFP Frame with the first packet, so the SFP Frame containing the first packet is finished.
2. The second packet in the VoQ is large enough that it must be segmented in order for it to be carried across the fabric. Its first segment becomes the only packet in the second SFP Frame. Its second segment becomes the first packet in the next SFP Frame.
3. The third packet in the VoQ fits completely into the rest of the SFP frame started by the second segment of the second packet. The third packet becomes the second packet in the third SFP Frame.



Note this Multiplexing algorithm does not affect the basic Segmentation/Reassembly process. In other words, a given application packet will be segmented the same way regardless of any SFP frame multiplexing that occurred on the packets that preceded it. So even though Multiplexing and Segmentation/Reassembly may occur within the same SFP frame (e.g. SFP Frame 3 in the above diagram), the Multiplexing and Segmentation processes may be decoupled in the implementation if desired.

All of this can be boiled down to one rule:

The source node allows the multiplexing of the ending segment of a packet with other small packets, as long as the ending segment is the 1<sup>st</sup> packetlet in the SFP frame. No other combinations of Segmentation and Multiplexing are allowed.

However you choose to think of it (algorithm or rule), the scheme described above allows a given implementation to decouple the Multiplexing and Segmentation processes if desired, as shown in the 2<sup>nd</sup> figure of section 3.3.1.

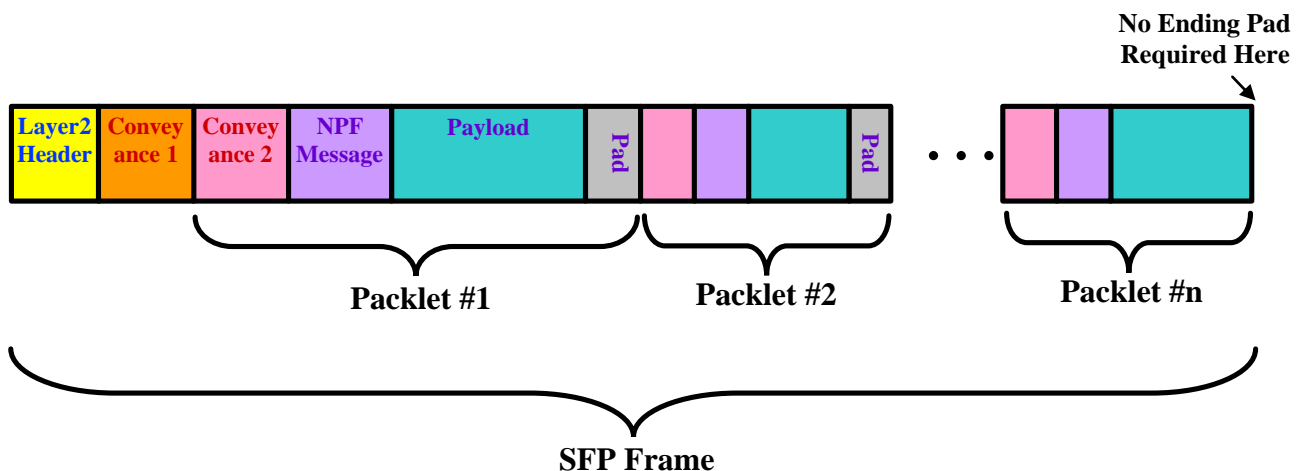
Note that the example in section 3.3.1 is not the only recommended implementation of SFP. It's just that the SFP multiplexing method described above is flexible enough to accommodate this type of implementation, as well as other implementations.

The bottom line is this: segmentation and multiplexing are specified such that the implementation of these two features can be implemented separately.

### 3.5.1 Packetlet Ending Pad

If an SFP frame is multiplexed, the source node adds padding as necessary to the end of each packetlet so as to align the next packetlet header in the frame on a 64-bit boundary.

Note that, since there is no "next packetlet header in the frame" after the last packetlet, no padding is required after the last packetlet. Similarly, when there is only one packetlet in the frame (i.e. non-multiplexed frame), no Packetlet Ending Pad is required.



Refer to the full specification of SFP.0 for further details on the Packetlet Ending Pad.

## 4 APPENDIX

This section adds additional explanatory information.

### 4.1 MULTIPLEXING TRADE-OFF ANALYSIS

The following examples use SFP over Ethernet/VLAN fabric.

The overhead for Ethernet with VLAN Fabric is:

Field	# of bytes
Preamble	7
Start of Frame	1
Mac Header	14
VLAN Header	4
Frame Check Sequence	4
Inter-frame gap	12
Ethernet with VLAN	42

The total fabric overhead may be expressed as the SFP Frame overhead plus the SFP Packlet overhead.

The SFP Frame overhead is the Ethernet with VLAN overhead + the SFP Conveyance Header 1 size.

Field	# of bytes
Ethernet with VLAN	42
SFP Conveyance Header 1	6
SFP Frame Overhead	48

This overhead figure of 48 bytes is used in the 2 sections below for comparison purposes.

The SFP Packlet overhead is 8 bytes (i.e. SFP Conveyance Header 2 plus NPF UID field).

### 4.1.1 TCP ACK packets over POS

This example assumes that a Packet Over Sonet (POS) I/O port is receiving back-to-back TCP ACK packets. A TCP ACK packet consists of

Header	# of bytes
POS Header, Flags & CRC	8
IP Header	20
TCP Header	20
TCP ACK Length	48

The POS Layer2 overhead is stripped off and the remaining 40 bytes are sent over the SFP Fabric (i.e. routing application). For this example, let's assume that 10 TCP ACK packets are sent back-to-back over the POS interface. This corresponds to 480 bytes over the POS interface. These 480 bytes get expanded over the Fabric, as shown in the spreadsheet below.

<u>Over Fabric Without Multiplexing</u>		<u>Over Fabric With Multiplexing</u>	
SFP Frame Overhead	48	SFP Packet Overhead	8
SFP Packet Overhead	8	TCP ACK Payload	40
TCP ACK Payload	40	Sum Above	48
Sum Above	96	x10 packetlets	480
x10 packets	960 bytes	+SFP Frame Overhead	48
			528 bytes

The amount of overhead for Fabric headers versus POS I/O headers is calculated below.

Without Multiplexing $960/480 = 2$ ( <b>100% overhead</b> )	With Multiplexing $528/480 = 1.1$ ( <b>10% overhead</b> )
--	--

To further illustrate the issue with TCP ACK packets over POS, consider the case of a line card with an OC-12 (i.e. 622 Mbits/sec) interface connected to a Gigabit Ethernet Fabric. Without SFP multiplexing, this scenario would not be viable, as back-to-back TCP ACK packets would double the bandwidth to over 1.2 Gbits/sec. However, with SFP multiplexing, the data rate would fit comfortably into a Gigabit Ethernet fabric, even with back-to-back TCP ACKs.

### 4.1.2 Wireless UMTS AMR Voice Packet

This example assumes that compressed voice packets using the AMR codec are being sent back-to-back over the wireless airwaves. The over-the-air bandwidth is generally used to specify the amount of wireless traffic. Since AMR voice packets are aggregated over the air, the specific amount of header overhead is approximated below.

Field	# of bytes
Total Wireless Headers (approximate)	2
AMR voice payload	12
Total AMR packet length	14

For this example, let's assume that 10 AMR Voice packets are sent back-to-back over the air. This corresponds to 140 bytes over the air. These 140 bytes get expanded over the Fabric, as shown in the spreadsheet below.

<u>Over Fabric Without Multiplexing</u>		<u>Over Fabric With Multiplexing</u>	
SFP Frame Overhead	48	SFP Packet Overhead	8
SFP Packet Overhead	8	AMR Voice Packet	14
AMR Voice Packet	14	Packet Ending Pad	2
<b>Sum Above</b>	<b>70</b>	<b>Sum Above</b>	<b>24</b>
x10 packets	<b>700 bytes</b>	x10 packets	240
		+SFP Frame Overhead	48
			<b>288 bytes</b>

The amount of overhead for Fabric headers versus Wireless over-the-air headers is calculated below.

Without Multiplexing $700/140 = 5$ ( <b>400% overhead</b> )	With Multiplexing $288/140 = 2.06$ ( <b>106% overhead</b> )
--	--

## 4.2 SFP Fault Tolerance Examples

This section is intended to explore the SFP issues that can arise from certain example redundancy or fault tolerance schemes. For simplicity, all example redundancy schemes discussed here have 2 completely separate Ethernet fabrics. Each node has 2 separate NIC interfaces, one for each fabric. Note that SFP will support other types of fault tolerance schemes, but the discussion in this section will be limited to a few commonly used redundancy schemes.

### 4.2.1 Active / Active

In this redundancy scheme, both fabrics are used actively with different flows being load balanced across the 2 fabrics. If one fabric fails, all connections over the faulty fabric must be moved to the other (i.e. still functional) fabric.

Since this scheme has two different active paths, there needs to be 2 different SFP.0 Flow Bundle values assigned. Otherwise, the destination would not have a workable context to check the received sequence numbers, segmentation/reassembly state, etc.

When a fault occurs, the connections have to be moved from one Flow Bundle to another. In addition, the Ethernet/MAC addresses may also be different for the 2 fabrics, so this may have to be changed on a fault as well.

The major advantage of this redundancy scheme is that the full bandwidth of both fabrics is usually available for bursty data. The main disadvantage is that the amount of time to move connections on a fault is generally much longer.

### 4.2.2 Active / Standby – Destination Select

In this scheme, the source node always transmits the same data redundantly over both fabrics. All destination nodes listen only to one fabric (i.e. the one that is deemed active). When a fault occurs on the active fabric, the destination nodes switch and listen to the other (previously standby) fabric.

Since the destination node only listens to one fabric at a time, the same Flow Bundle value may be used across both fabrics. This would allow flows (i.e. UID values) to retain the same Flow Bundle value after a failover. However, depending on how the Ethernet/MAC addresses are allocated, the Ethernet/MAC addresses may be different for the 2 fabrics, which means that the source and destination MAC address for a given UID flow may be different after a failover. Alternatively, if the endpoint nodes support an option to use the same Ethernet MAC addresses for the redundant fabrics, then these addresses would not change after a failover.

### 4.2.3 Active / Standby – Source Select

In this scheme, the source node only transmits user data over one fabric (i.e. the active fabric). The standby fabric may have some health monitoring packets sent, but no real data. When a fault occurs on the active fabric, the source nodes switch and send all user data to the other (previously standby) fabric.

The destination node receives user data from only one fabric at a time, except possibly for a very short time during a failover switch. For this reason, the same Flow Bundle value may be used across both fabrics. This would allow flows (i.e. UID values) to retain the same Flow Bundle value after a failover.

However, depending on how the Ethernet/MAC addresses are allocated, the Ethernet/MAC addresses may be different for the 2 fabrics, which means that the source and destination MAC address for a given

UID flow may be different after a failover. If the MAC addresses are different, the Flow Bundle structures (e.g. virtual output queue structures) at the source node must be updated to use the new Ethernet addresses, otherwise the packets won't be delivered by the fabric.

Alternatively, if the endpoint nodes support an option to use the same Ethernet MAC addresses for the redundant fabrics, then these addresses would not change after a failover.