Abstract

Generic Framing Procedure (GFP) was developed to allow efficient transport of packet data through SONET/SDH networks, making use of the new virtual concatenation and LCAS technologies for creating flexible-sized transport channels. Although a competing approach, explored by some start-up carriers, attempted to use a pure packet network (e.g., an Ethernet WAN), the conversion from a SONET/SDH backbone network to an Ethernet backbone would have been cost-prohibitive and would not have provided the desired integrated OAM&P capabilities. GFP technology allowed efficient packet transport within the existing SONET/SDH backbone and, thus, is very attractive to carriers.

This white paper details the GFP technology in depth, as it is currently not covered in any existing textbooks.

About the Author

Steve Gorshe, Ph.D. is a Principal Engineer in the Product Research Group and oversees ICs for SONET, optical transmission and access systems.

Currently Steve is a senior member of the IEEE and co-editor for the IEEE Communications magazine’s Broadband Access Series. He is the chief editor for the ANSI T1X1 Subcommittee, which is responsible for SONET and optical network interface standards. He is a recent recipient of the Committee T1 Alvin Lai Outstanding Achievement Award for his standards work and has been a technical editor for T1.105, T1.105.01, T1.105.02, and T1.105.07 within the SONET standard series as well as the ITU-T G.7041 (GFP) G.7043 (Virtual concatenation of PDH signals), G.8040 (GFP mapping into PDH signals), and G.8011.1 (Ethernet Private Line Service) recommendations. He has 26 patents issued or pending and several published papers.

Revision History

<table>
<thead>
<tr>
<th>Issue No.</th>
<th>Issue Date</th>
<th>Details of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>April 2005</td>
<td>Enhanced and updated with new material developed over the last few years.</td>
</tr>
<tr>
<td>1</td>
<td>July 2004</td>
<td>Document created</td>
</tr>
</tbody>
</table>
Contents

Abstract.............................................................................................................................. 1
About the Author...............................................................................................................1
Revision History................................................................................................................1
Contents............................................................................................................................. 2
List of Figures ................................................................................................................... 3
List of Tables ..................................................................................................................... 4
Preface ............................................................................................................................... 5
1 Introduction ............................................................................................................... 6
   1.1 GFP background ................................................................................................. 6
2 GFP frame structure.................................................................................................. 7
3 Frame mapped GFP (GFP-F).................................................................................... 9
4 Transparent GFP (GFP-T)....................................................................................... 13
5 Performance Considerations and Client Management Frames ......................... 20
6 Error handling considerations............................................................................... 21
7 Conclusions............................................................................................................. 22
8 References............................................................................................................... 23
9 Notes ........................................................................................................................ 25
List of Figures

Figure 1  GFP frame structure ......................................................................................... 7
Figure 2  Example client data mappings into GFP-F ....................................................... 9
Figure 3  64B/65B block code structure......................................................................... 14
Figure 4  Example of mapping client data and control characters into a 64B/65B code ........................................................................................................... 15
Figure 5  GFP-T superblock structure............................................................................ 16
Figure 6  Construction of the GFP-T frame.................................................................... 17
Figure 7  Example of 65B_PAD insertion ...................................................................... 18
Figure 8  Sub-rate GFP-T example for Fibre Channel................................................... 19
List of Tables

Table 1  List of client payloads mapped into GFP-F (accepted and proposed) .......... 10
Table 2  List of client payloads mapped into GFP-T..................................................... 13
Table 3  Virtually concatenated channel sizes for various transparent GFP clients 17
Preface

The goal is to provide an overview of GFP for a reader who is unfamiliar with this technology. To this end, a brief introduction is provided that includes the motivations that led to the development of the GFP standard. The remainder of the white paper presents various aspects of the GFP frame structure, the mapping of client data frame or signals into GFP, and GFP performance or performance monitoring considerations.

The information in this white paper has also been adapted to form part of the following textbook: M. Elanti, S. Gorshe, L. Raman, and W. Grover, *Next Generation Transport Networks – Data, Management, and Control Plane Technologies*, Springer, 2005.
1 Introduction

GFP was developed to allow efficient transport of packet data through SONET/SDH networks, making use of the new virtual concatenation and LCAS technologies for creating flexible-sized transport channels. A competing approach that was explored by some start-up carriers was to use a pure packet network (e.g., an Ethernet WAN). The conversion from a SONET/SDH backbone network to an Ethernet backbone would not only have been cost prohibitive, but data networks such as Ethernet typically have not provided the integrated Operations, Administration, Maintenance and Provisioning (OAM&P) capabilities that have made SONET/SDH so valuable to carriers. A technology such as GFP that allowed efficient packet transport within the existing SONET/SDH backbone was thus very attractive. This white paper covers the GFP technology in depth, as it is currently not covered in any existing textbooks.

1.1 GFP background

GFP was originally born in T1X1 Technical Subcommittee to address some limitations of ATM and Packet over SONET (PoS). Subsequently, in order to expedite international agreement it was decided to have the International Telecommunication Union -Telecommunication Standardization Sector (IUT-T) publish the final standard. GFP is now specified in ITU-T Recommendation G.7041.

As described in PMC-Sierra’s white paper, A tutorial on SONET/SDH (PMC-2030895), ATM suffers from bandwidth inefficiency due to the amount of cell overhead relative to payload capacity. Another perceived drawback to ATM is that ATM implementations typically include a great detail of complexity that is not required for relatively simple packet data interconnections (e.g., the ATM signaling protocols). The drawbacks to PoS are its non-deterministic bandwidth requirements and its requirement that the client data packets be mapped into the IETF Point-to-Point Protocol (PPP) and HDLC for Layer 2. The first drawback is a consequence of the requirement that escape characters be added for each client data byte that looks like an HDLC control character. In the worst case (a situation that could potentially be created by a malicious user), the byte-stuffed HDLC packet payload field could be twice as long as the original payload. The requirement to map into PPP is a distinct drawback if a carrier wants to offer a simple Ethernet extension. Here, the Ethernet MAC frame would need to be terminated so that the client data could be re-mapped into PPP with HDLC and a new Ethernet MAC frame generated on the other end of the PoS link. Despite the drawbacks, with over 90% of the data transported through the public WAN originating and/or terminating as Ethernet at the customer premises, Ethernet extension is a naturally desirable service for the carriers to provide.

GFP allows the simple, efficient transport of a variety of data signals through SONET/SDH, asynchronous/PDH, and G.709 OTN networks. The protocol and its capabilities are explained in Section 2.
2 GFP frame structure

The basic structure of a GFP frame is illustrated in Figure 1. The four-byte Core header consists of a 16-bit Payload Length Indicator (PLI) with a CRC-16 to protect the PLI. The PLI is the binary count of the number of bytes in that GFP frame’s payload field. The Core header is the key to eliminating the bandwidth expansion problem of PoS. Rather than relying on a special character for frame delimiting, the GFP receiver begins the framing process by looking for a 16-bit field that is followed by a correct CRC-16 for that field. When such a 32-bit pattern is found, it is very likely to be the Core header at the beginning of a GFP frame, as the probability of such a pattern randomly occurring in the client data is $2^{-32}$. The GFP receiver then uses the PLI information to determine where the end of the GFP frame is. The next Core header will occur immediately after the end of this current frame, so if another valid 32-bit pattern is found in that location, the receiver can be sure that it has acquired framing for the GFP stream. The CRC-16 is adequate to provide a simple, single error correction capability for the PLI, which increases the robustness of the GFP framing. (In contrast, an error in an HDLC start/end flag has no error correction, which makes HDLC framing more vulnerable to transmission errors.)

Figure 1  GFP frame structure
The GFP payload area consists of payload header information, a payload field that contains the client data, and can optionally contain a CRC-32 over the payload field. The payload headers are broken down into two types. The payload Type header is used in all GFP frames except Idle frames, and consists of 2 bytes of overhead information protected by a CRC-16. If additional payload overhead information is needed (e.g., a channel identifier to identify different GFP connections when GFP frames from different sources are frame-multiplexed together), then an Extension may also be used. Specifically, the Type header contains a Payload Type Indicator (PTI), an indication of whether the optional payload frame check sequence is included in the payload field (PFI), an indication of what type of Extension header, if any, is present (EXI), and a User Payload type indicator (UPI). The only PTI types identified so far are client data frames and client management frames (CMFs). For the client data frames, the UPI indicates the type of payload that is mapped into the GFP frame, and whether that mapping is frame-based or transparent. The only Extension header defined at this time is the Linear Extension header, which consists of an 8-bit channel identification field, an 8-bit spare/reserved field, and a CRC-16 over the header. Other Extension headers are under consideration that would potentially expand the identification field so that flows within channels could be identified.

Any idle time between GFP frames carrying client data is filled with GFP Idle frames. A GFP Idle frame consists of only a Core header with a PLI=0.
3 Frame mapped GFP (GFP-F)

The initial GFP work focused on GFP-F. The basic concept is to take a client frame (typically a Layer 2 frame), remove its unnecessary overhead (e.g., frame delimiting characters or preamble), and then simply place the remainder of the client data frame into the GFP Payload Field. Hence, there is a one-to-one mapping between client data frames and GFP-F frames. An example of the Ethernet mapping is shown in Figure 2(a). The specific bit alignments between the client data, the GFP-F frame, and the SONET/SDH payload are illustrated in the PMC-Sierra white paper [4].

Figure 2 Example client data mappings into GFP-F
Note that for the Ethernet example of Figure 2(a), it is assumed that the optional payload FCS (pFCS) is not used.\(^1\) When the client data frame is known to have its own frame FCS, the GFP payload FCS is somewhat redundant since a GFP-F demapper will typically have the capability to examine the client data FCS. The GFP payload FCS would only be used if the client data frame either had no FCS of its own or a weaker FCS (e.g., a CRC-16).

The simplicity of the GFP-F mapping is a key to GFP’s flexibility. While most of the initial focus on GFP-F was for carrying Ethernet payloads, there is growing interest in using GFP-F for other payloads. As a result, there have been several new mappings proposed. The current mappings into GFP-F are listed in Table 1.

### Table 1  List of client payloads mapped into GFP-F (accepted and proposed)

<table>
<thead>
<tr>
<th>GFP-F client payload type</th>
<th>UPI value &lt;7:0&gt; (PTI = 000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>0000 0001</td>
</tr>
<tr>
<td>PPP</td>
<td>0000 0010</td>
</tr>
<tr>
<td>Multiple Access Protocol over SDH (MAPOS)</td>
<td>0000 1000</td>
</tr>
<tr>
<td>IEEE 802.17 Resilient Packet Ring</td>
<td>0000 1010</td>
</tr>
<tr>
<td>Fibre Channel FC-BBW</td>
<td>0000 1011</td>
</tr>
<tr>
<td>MPLS direct mapping</td>
<td>0000 1101</td>
</tr>
<tr>
<td>MPLS (multicast)</td>
<td>0000 1110</td>
</tr>
<tr>
<td>IS-IS</td>
<td>0000 1111</td>
</tr>
<tr>
<td>IPv4</td>
<td>0001 0000</td>
</tr>
<tr>
<td>IPv6</td>
<td>0001 0001</td>
</tr>
<tr>
<td>PPP direct mapping</td>
<td>(note)</td>
</tr>
<tr>
<td>HDLC</td>
<td>(note)</td>
</tr>
<tr>
<td>Reserved for proprietary use.</td>
<td>1111 0000 through 1111 1110</td>
</tr>
</tbody>
</table>

Note:

- A proposal under active consideration is to have a PPP mapping that omits the HDLC header and replace the HDLC FCS with a GFP pFCS in order to simplify processing and improve efficiency. As part of this discussion, there has been a proposal to add a generic mapping for any HDLC-framed client.

\(^1\) In fact, ITU-T Recommendation G.8012 (2004), Ethernet UNI and Ethernet NNI specifies that the GFP pFCS is not used when Ethernet is carried in GFP-F.
A PPP mapping was part of the original G.7041 standard. (See [15] for the complete definition of PPP.) As seen in Figure 2(b), this mapping includes the HDLC-like framing overhead specified in IETF RFC 1662 (“PPP in HDLC-like Framing”), to be consistent with the PPP over SONET/SDH (PoS) mapping (see [4]). In PoS, the HDLC flags are also retained so that the HDLC framing can be used for frame delineation. The HDLC Address and Control bytes are defined in RFC 1662 to carry 1111 1111 and 0000 0011, respectively, so for the GFP mapping, the HDLC header contains no useful information. It is possible to use a PPP Link Configuration Protocol (LCP) optional procedure to remove the HDLC header and replace the PPP type field with a 1-2 byte PPP protocol field. There is also a proposal under active consideration to have an optimized PPP mapping into GFP-F that always removes the HDLC header and FCS, and uses the GFP pFCS for error detection. At the same time, there has been discussion about whether there are enough applications to justify a generic HDLC into GFP-F mapping. At the time of this printing, these topics are on the G.7041 “Living List” for further study.

GFP-F was originally defined as a Layer 1 (1.5) transport encapsulation mechanism for carrying Layer 2 frames. It was appreciated at the time of its development that the existence of the PTI and UPI fields along with the channel identification field in the linear Extension Header gave GFP a rudimentary set of Layer 2 type capabilities. There had been some reluctance to exploit (or enhance) these features since there was general agreement that GFP should not be ballooned in complexity to become a full switching layer similar to ATM. On the other hand, it made sense to exploit these capabilities to improve network bandwidth efficiency.

Recently, some of these GFP Layer 2–like capabilities have been exploited for Multiprotocol Label-Switching (MPLS) frame transport (see [20], [21], and [22]). MPLS adds a header to a client data packet that contains a label field to identify what is called a forwarding equivalency class (FEC). MPLS label-switched routers (LSRs) use these labels to route packets through the MPLS network. All packets sharing the same label value are then treated in the same manner by a LSR. The MPLS labels are not global, but are assigned with local significance between two adjacent LSRs. IP or IS-IS frames are typically used to communicate the information required to establish the MPLS label assignments among MPLS network nodes and the hop-by-hop routes that will be taken for packets with each MPLS label. MPLS, which is nominally somewhere between a Layer 3 and Layer 2 protocol, typically also uses a Layer 2 such as PPP. The Layer 2 protocol provides a protocol identifier field that identifies its payload as being either MPLS or IP (or other protocol) frames and an error check over the frame. At the request of multiple carriers, a

2 While some have seen particular advantages to being able to send GFP frames along a pre-provisioned route based on the ID fields in an Extension header, no one wanted to create the equivalent of an ATM switched virtual circuit (SVC) network with its own signaling protocol, etc. One of the proposals on the current G.7041 Living List is a new linear Extension Header with an expanded address (flow ID) field and other fields similar to an MPLS header for this pre-provisioned routing. The intention was to allow mapping MPLS header information into the GFP frame to allow the transport provider to route frames without having to go up to the MPLS layer. It would also allow simultaneous routing of non-MPLS packets on the same link. The future of this proposal is uncertain.

3 The Label Distribution Protocol (LDP) and Resource Reservation Protocol for Traffic Engineering (RSVP_TE) are the two protocols outlined in IETF RFCs 3031 and 3270, respectively, for communicating the label assignments among MPLS LSRs. Other, non-IP-based protocols (e.g., IS-IS) have also been used by some vendors and networks.
direct mapping of MPLS into GFP with no Layer 2 protocol overhead bytes was added, as shown in Figure 2(c). These carriers look to MPLS as a key technology for their core networks. At the same time, they are concerned about bandwidth efficiency and about reducing the number of layers/protocols that need to be provisioned and administered in their networks. The GFP PTI/UPI codes identify the MPLS payload, and the GFP pFCS provides the frame error check. To complement direct MPLS mapping, direct mappings into GFP-F were also added for IP (IPv4 and IPv6) and IS-IS in order to carry the MPLS control plane information. These mappings are essentially the same as the MPLS mapping (i.e., they use the GFP pFCS for their error detection). With the addition of these mappings, Layer 2 can become a null layer when MPLS is carried over GFP. It should be noted that although direct IP mappings into GFP-F were primarily added to support MPLS control plane traffic, they could also be used as a generic mapping for IP networks. Some carriers are considering this possibility.
4 Transparent GFP (GFP-T)

Several important data protocols use the 8B/10B line code at the physical layer. The protocols and their associated UPI codes are shown in Table 2. Fibre Channel, Fibre Connection (FICON), and Enterprise System Connection (ESCON) are commonly used in modern storage area networks (SANs) and Digital Video Broadcast – Asynchronous Serial Interface (DVB-ASI) is popular for video distribution. The 8B/10B line code maps the 2^8 possible 8-bit byte values into one or two of the 2^{10} possible 10-bit values. One goal of this mapping is to maintain a running balance between the number of 1s and 0s that are transmitted. Since only a limited number of 10B characters can have five 1s and five 0s, a given 8-bit data value is mapped into two different 10B values, each with a complementary number of 1s relative to the other. The difference in this running 0/1 balance at the end of each transmitted 8B/10B character is referred to as the running disparity, and it is used to choose which of the two potential 10B values is used for next 8B data value to be transmitted. Each of the above protocols exploits the fact that there are leftover 10B characters that can be used as control codes (e.g., to indicate the start and end of a data frame or to trigger some type of link initialization). For some protocols, a control code flags the beginning of a sequence of characters that communicate control information. Such sequences are called primitive sequences.

<table>
<thead>
<tr>
<th>GFP-T client payload type</th>
<th>UPI value (&lt;7:0&gt;) (PTI = 000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fibre Channel</td>
<td>0000 0011</td>
</tr>
<tr>
<td>FICON</td>
<td>0000 0100</td>
</tr>
<tr>
<td>ESCON</td>
<td>0000 0101</td>
</tr>
<tr>
<td>Gbit Ethernet</td>
<td>0000 0110</td>
</tr>
<tr>
<td>DVB-ASI</td>
<td>0000 1001</td>
</tr>
<tr>
<td>Asynchronous Fibre Channel (see note)</td>
<td>0000 1100</td>
</tr>
</tbody>
</table>

Note:

- Asynchronous Fibre Channel is asynchronous in the sense that the transport container is not required to have as much capacity as the Fibre Channel interface.

If the frames from these protocols were carried with GFP-F, the 8B/10B control code information would be lost. Another consideration for the SAN protocols is that they are typically very sensitive to transmission delay (latency) between the two SAN nodes. The one drawback to using the Core header with its PLI for GFP-F mappings is that an entire client data frame must be buffered prior to transmitting the GFP-F frame in order to determine the required PLI value for that frame. In most applications, these issues are not a concern. For these 8B/10B-encoded clients, however, the control code transparency and latency issues can become important. A tribute to the versatility (“generic-ness”) of the GFP protocol is that a variation of the protocol was developed that resolved these transparency and latency issues while providing a significant transmission bandwidth savings over what would be required to send a stream of the native 8B/10B characters. Due to its transparency to 8B/10B codes, this GFP mapping came to be called transparent GFP (GFP-T).
The desire for both transmission bandwidth efficiency and the transparent communication of the control codes was realized by transcoding the character stream into a new block code. The first step in the GFP-T mapping process is to decode the 8B/10B characters into their original 8-bit data values or the associated control characters. These data and control characters are then re-encoded into a 64B/65B code. Each 64B/65B code carries the information of eight of the 8B/10B characters. For data characters, the original 8-bit data values are mapped directly into the payload bytes of the 64B/65B codes. The 8B/10B control codes are re-encoded and mapped into their own bytes. The details of the 64B/65B code construction are illustrated in Figure 3. The first bit of the 65B block indicates whether there were any 10B control codes among the 8 characters encoded into that 64B/65B code. If control codes are present, they are placed in the first byte(s) of the 64B/65B code payload. Since there are 12 or fewer control codes used for any 8B/10B client, a 4-bit code is adequate to represent them. A 3-bit address indicates where the control code occurred in the client character stream relative to the other characters encoded into that 64B/65B block. The MSB of a control code byte indicates whether this is the last control code in that 64B/65B block or whether the next byte also contains a control code. Figure 4 gives an example of the mapping of data and control characters (designated as D and K characters, respectively) into a 64B/65B code.

**Figure 3** 64B/65B block code structure

<table>
<thead>
<tr>
<th>Input Client Character</th>
<th>Flag Bit</th>
<th>64-Bit (8-Octet) Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Octet 0</td>
<td>Octet 1</td>
</tr>
<tr>
<td>All data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 data, 1 control</td>
<td>0</td>
<td>D1</td>
</tr>
<tr>
<td>6 data, 2 control</td>
<td>1</td>
<td>0 aaa C1</td>
</tr>
<tr>
<td>5 data, 3 control</td>
<td>1</td>
<td>1 aaa C1, 1 bbb C2</td>
</tr>
<tr>
<td>4 data, 4 control</td>
<td>1</td>
<td>1 aaa C1, 1 bbb C2, 1 ccc C3</td>
</tr>
<tr>
<td>3 data, 5 control</td>
<td>1</td>
<td>1 aaa C1, 1 bbb C2, 1 ccc C3, 1 ddd C4</td>
</tr>
<tr>
<td>2 data, 6 control</td>
<td>1</td>
<td>1 aaa C1, 1 bbb C2, 1 ccc C3, 1 ddd C4, 1 eee C5</td>
</tr>
<tr>
<td>1 data, 7 control</td>
<td>1</td>
<td>1 aaa C1, 1 bbb C2, 1 ccc C3, 1 ddd C4, 1 eee C5, 1 fff C6</td>
</tr>
<tr>
<td>8 control</td>
<td>1</td>
<td>1 aaa C1, 1 bbb C2, 1 ccc C3, 1 ddd C4, 1 eee C5, 1 fff C6, 1 ggg C7, 1 hhh C8</td>
</tr>
</tbody>
</table>

Legend:
- Leading bit in a control octet (LCC) = 1 if there are more control octets and = 0 if this payload octet contains the last control octet in that block
- aaa = 3-bit representation of the 1st control code’s original position (1st Control Code Locator)
- bbb = 3-bit representation of the 2nd control code’s original position (2nd Control Code Locator)
- hhh = 3-bit representation of the 8th control code’s original position (8th Control Code Locator)
- Ci = 4-bit representation of the i<sup>th</sup> control code (Control Code Indicator)
- Di = 8-bit representation of the i<sup>th</sup> data value in order of transmission
The 64B/65B codes have two drawbacks, however. The first is that the 65-bit code length would prevent a stream of 64B/65B characters from being byte-aligned with the SONET payload bytes. Byte alignment, which is a feature of all the other SONET/SDH payload mappings, has the advantages of allowing direct payload data byte observability within the SONET/SDH stream and of allowing the use of parallel data path implementations. Parallel data paths within devices and systems allow lower device/system clock rates. The lower clock rates in turn allow the use of more power-efficient technology such as CMOS. The second drawback of the 64B/65B code is its relative lack of error detection capability. (Error considerations are treated in greater detail below.) Both of these drawbacks were addressed by the combining of eight 64B/65B codes into a superblock with an appended error check. The superblock structure is illustrated in Figure 5. The payload data bytes of the eight constituent 64B/65B codes are placed into the superblock in transmission order, with the eight leading (flag) bits of these codes grouped together in a trailing byte. Two more trailing bytes are then added as a CRC-16 error check code over the information in that superblock.
Figure 5 GFP-T superblock structure

where: Octet $j, k$ is the $k^{th}$ octet of the $j^{th}$ 64B/65B code in the superblock
L$j$ is the leading (Flag) bit $j^{th}$ 64B/65B code in the superblock
C$i$ is the $i^{th}$ error control bit

The final step in the GFP-T mapping is to insert an integer number of superblocks into a GFP frame. The process of going from the 64B/65B code to a GFP frame is illustrated in Figure 6. The GFP-T frame is identical to a GFP-F frame except for the specific PTI and UPI values that are used in the Type header. Since GFP-T has CRC-16 error checks per superblock, the payload FCS is typically not used. The number of superblocks that are grouped into a GFP-T frame is a function of the difference between the client data rate, the channel rate of the SONET/SDH channel, and the amount of “spare” bandwidth that is desired for client management frames. The minimum number of superblocks for various clients is shown in Table 3. The details for calculations regarding the number of superblocks and the resulting spare bandwidth are shown in Corrigendum 1 of G.7041.
Figure 6  Construction of the GFP-T frame

Table 3  Virtually concatenated channel sizes for various transparent GFP clients

<table>
<thead>
<tr>
<th>Client Signal</th>
<th>Nominal Native (Unencoded) Client Signal Bandwidth</th>
<th>Minimum Virtually-Concatenated Transport Channel Size</th>
<th>Nominal Transport Channel Bandwidth</th>
<th>Minimum Number of Superblocks per GFP Frame</th>
<th>Worst/ Best-case Residual Overhead Bandwidth</th>
<th>Best case client management payload bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fibre Channel</td>
<td>850 Mbit/s STS-3c-6v / VC-4-6v</td>
<td>898.56 Mbit/s</td>
<td>13</td>
<td>412 Kbit/s / 85.82 Mbit/s</td>
<td>2.415 Mbit/s</td>
<td>2.415 Mbit/s</td>
</tr>
<tr>
<td>Gbit Ethernet</td>
<td>1.0 Gbit/s STS-3c-7v / VC-4-7v</td>
<td>1.04832 Gbit/s</td>
<td>95</td>
<td>281 Kbit/s / 1.138 Mbit/s</td>
<td>376.5 kbit/s</td>
<td>376.5 kbit/s</td>
</tr>
</tbody>
</table>

NOTES

1. The worst-case residual bandwidth occurs when the minimum number of superblocks is used per GFP frame. The best case occurs for the value of N that allows exactly one Client Management frame per GFP data frame. A 160-bit Client Management frame was assumed for the best case (with a CRC-32). For both cases, it was assumed that no Extension headers were used.
2. The best-case client management payload bandwidth assumes 8 “payload” bytes per Client Management frame and the best-case residual overhead bandwidth conditions.

3. Fibre Channel also supports rates of 425, 1700, and 3400 Mbit/s (unencoded). The SONET/SDH channel size scales linearly for these rates, and the minimum number of superblocks is 13 for all cases.

Latency reduction was another consideration for GFP-T. Since GFP-T encapsulates a number of client data characters rather than a whole client data frame there is no implied correlation between the boundaries of the GFP-T frames and the client data frames. As a result, it would appear that the GFP frame PLI value could be pre-determined, thus eliminating the need to buffer an entire GFP frame of data prior to transmitting it. The problem, however, is that the SONET channel necessarily has a slightly higher bandwidth than is needed to carry the client data in the GFP-T stream. The client signal and the SONET/SDH VCAT channel rates required to carry the full client signal rate are also shown in Table 3. The higher bandwidth of the transmission channel means that the ingress buffer at the GFP-T mapper will periodically underflow as the mapper extracts bytes at a somewhat higher rate than the client data stream supplies them. Unless additional buffering (with its associated latency) was added at the mapper, the underflow situation would be a problem if the GFP frame were already being transmitted. In order to avoid this additional latency, a special 64B/65B control character was defined as a 65B_PAD that is inserted into a 64B/65B block code whenever the mapper ingress buffer approaches an underflow threshold. As illustrated in Figure 7, the 65B_PAD character is inserted into the 64B/65B code in exactly the same manner as if an 8B/10B control code had been received in the client data stream. The GFP-T demapper recognizes this character as dummy padding and discards it.

**Figure 7  Example of 65B_PAD insertion**

<table>
<thead>
<tr>
<th>Octet #</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Byte Stream</td>
<td>D1</td>
<td>K1</td>
<td>D2</td>
<td>buffer underflow</td>
<td>D3</td>
<td>K2</td>
<td>D4</td>
<td>D5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Octet #</th>
<th>L</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>65B Code Stream</td>
<td>1</td>
<td>1,001.C1</td>
<td>1,011.P1</td>
<td>0,101.C2</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
<td>D5</td>
</tr>
</tbody>
</table>

D = client data byte  
K = control character  
L = Leading 64B/65B bit  
P = 65B_PAD character

It is typically not cost effective for a customer to subscribe to a full-rate channel through the WAN (see Table 3). There are often idle periods between data packets such that the actual average client data rate is a fraction of the full rate. In order to allow a sub-rate connection while still preserving transparency to 8B/10B control codes, Committee T11, working with ITU-T SG15, has included an asynchronous, sub-rate GFP-T option as part of their new FC-BB-3 standard. The idea, illustrated in Figure 8, is to have a circuit between the full-rate client signal interface and the GFP-T mapper. This circuit removes idle characters from the full-rate stream and outputs a sub-rate 8B/10B encoded stream to the GFP-T mapper. Similarly, a circuit inserts idle characters into the sub-rate stream output from the GFP-T demapper in order to create the
full-rate egress stream. The sub-rate would correspond to the desired WAN channel rate (e.g., DS3 or STS-3). Due to the need to buffer client packets for the process of removing the idle characters, there is an increased latency with sub-rate GFP-T over that of full-rate GFP-T. The latency of sub-rate GFP-T is still typically less than that required for a GFP-F mapping.

**Figure 8** Sub-rate GFP-T example for Fibre Channel
5 Performance Considerations and Client Management Frames

At the time of writing, the OAM aspects of GFP are being defined. The GFP demapper can check such parameters as errors detected in any of the GFP headers, a mismatch from the expected values in the payload Type (or Extension) header, errors in the payload field (if the optional GFP payload FCS is used), and loss of GFP frame synchronization. These faults are reported to the network management system by the NE containing the GFP demapper. Some carriers have indicated a desire to perform single-ended performance monitoring similar to SONET/SDH paths/trails; i.e., have the far end report its performance parameters so that each end knows both the parameters that it calculates directly and the parameters calculated by the far end. If this capability is adopted for GFP, the demapper reports will be sent in GFP Client Management Frames. CMFs use a different PTI value and their own set of UPI values, but are otherwise identical to the GFP client data frames. The one application that is currently standardized for CMFs is the client-signal-fail indication that is discussed in the next section.
6 Error handling considerations

Transmission errors can occur on the client data stream either outside or within the SONET/SDH network. GFP-F and GFP-T handle these errors and faults differently due to the manner in which they perform the encapsulation process.

Client frame delimiting is part of the GFP-F mapping process, and a typical part of frame delimiting is checking the client data frame’s frame check sequence. If this check is bad, the GFP mapper discards that client data frame rather than encapsulating it. Similarly, if the GFP-F demapper receives a GFP frame with a bad check sequence in the any of the GFP headers, the GFP payload FCS (if it is being used), or the client data frame itself, that client data frame is discarded.

GFP-T focuses on the character level rather than the frame level. Since a limited number of 10-bit values are used for 8B/10B codes, and since the running disparity is updated with each character, transmission errors can typically be detected in the 8B/10B stream. When the GFP-T mapper detects either an illegal 8B/10B character or a running disparity error, it encodes the offending character as a special 10B_ERR control character in the 64B/65B code. When the GFP-T demapper decodes the 64B/65B code it encodes the 10B_ERR code into a particular illegal 8B/10B character so that the client receiver will be aware of the error.

Most of the redundant information that makes error detection possible is removed when the 8B/10B characters are transcoded into the GFP-T 64B/65B codes. The 64B/65B codes are particular vulnerable to errors in the leading (flag) bit or the last control code indicator bits if control codes are present. Errors in these particular bits can cause the GFP-T demapper to misinterpret multiple data/control characters, which could in turn create a burst error condition that is beyond the ability of the client frame check sequence (typically a CRC-32) to detect. The CRC-16 code that was added to each superblock provides a strong triple error detection and (optional) single error correction capability. When the GFP-T demapper sees a bad CRC-16 for a superblock, it treats all 64 client characters in that superblock as if they had been received as 10B_ERR codes. (Single errors can be simply corrected if that option is used.)

NOTE – The self-synchronous scrambler that is used on the GFP payload field inherently multiplies the transmission errors by creating a duplicate error 43 bits after the original error. This error multiplication would weaken the performance of each of the existing standard CRC-16 codes. As a result, a new CRC-16 was developed for the GFP-T superblock that preserved its error detection and correction capability in the presence of the descrambler, and was optimized for this specific block length. The interested reader can see [16] or [17] for more details.

In the case of a client signal failure prior to the GFP mapper, both the GFP-F and GFP-T mappers send a periodic client signal fail (CSF) frame to the GFP demapper. This signal failure could be the complete loss of the incoming signal or the loss of character synchronization for those clients using a character-oriented line code. The CSF message is sent in a Client Management Frame every 100 ms to 1000 ms. The action taken by a GFP demapper is client specific. In the case of GFP-T, the output 8B/10B stream will contain the same characters used to indicate a bad received 8B/10B character at the mapper. (Note that some client signals have other options that could be used in the situation.)
7 Conclusions

As data services increase in importance and volume through the public wide area networks, it has become important to adapt the existing SONET/SDH infrastructure to carry this traffic. GFP is one of the key enabling technologies for this adaptation. GFP provides a simple, flexible data encapsulation mechanism with a very robust frame delineation mechanism and a deterministic overhead bandwidth. The flexibility of GFP is already apparent from the number of client data signal and data frame mappings that have been defined for GFP. These mappings range from Layer 2 data frames, to Layer 3 packets, to Layer 1 character-oriented data streams. These factors have lead to GFP gaining broad global acceptance as the preferred data encapsulation mechanism for transport networks.

PMC-Sierra is a world leader in ICs supporting Ethernet mapping into SONET/SDH signals. The Arrow 2xGE (PM5397), which was the first commercially-available IC supporting GFP, maps two Gigabit Ethernet streams into an OC-48/STM-16. The Arrow 24xFE (PM5329) maps 24 10/100M Ethernet streams into an OC-12/STM-4 using GFP with either high or low order virtual concatenation. Arrow 8xFE (PM5333) provides the same functionality as the Arrow 24xFE for eight 10/100M Ethernet channels. The ADM-622 (PM5337) provides the functions of an add/drop multiplexer on a single chip, including the ability to map a single Gigabit Ethernet or up to eight 10/100M Ethernet streams into an OC-12/STM-4 with GFP.
8 References


[22] IETF RFC 3270, May 2002 “Multi-Protocol Label Switching (MPLS) support of Differentiated Services”

[23] ANSI INCITS xxx-2004⁴, Information Technology – Fibre Channel Backbone (FC-BB-3)

---

⁴ This standard was in the process of formal approval at the time of publishing, so the number was not available.
9 Notes