Project	IEEE 802.16 Broadband Wireless Access Working Group <**http://ieee802.org/16**>
Title	A Generic Packet Convergence Sublayer (GPCS) for Supporting Multiple Protocols over 802.16 Air Interface
Date Submitted	2005-11-17
Source(s)	Lei Wang, Brian Petry, Yair Bourlas, Kenneth Stanwood, Cygnus Communications Inc. Voice: 760-448-4168 Fax: 760-448-1989 lwang@cygnuscom.com bpetry@cygnuscom.com ybourlas@cygnuscom.com kstanwood@cygnuscom.com Phillip Barber, Huawei pbarber@broadbandmobiletech.com GuoQiang Wang, Mo-Han Fong, Nortel Networks guoqiang@nortel.com mhfong@nortelnetworks.com Mary Chion, ZTE San Diego marychion@ztesandiego.com
Re:	This is a follow-up contribution on the proposal of a generic packet convergence sublayer.
Abstract	As requested by 802.16g during the comments resolution discussions in session #39, we are submitting a follow-up contribution on our Generic Packet Convergence Sublayer proposal i.e., C802.16g-05/25, by presenting additional supporting materials and integrating suggestions and comments received from other members. We are concerned that the 802.16 protocol cannot easily be extendable to transport new protocols over the 802.16 air interface. It would appear that a convergence sublayer is needed for every type of protocol transported over the 802.16 MAC. Every time a new protocol type needs to be transported over the 802.16 air interface, the 802.16 standard needs to be modified to define a new CS type. We need to have a generic Packet convergence sublayer that can support multi-protocols and which does not require further modification to the 802.16 standard to support new protocols. We believe that this was the original intention of the Packet CS. Furthermore, we believe it is difficult for the industry to agree on a set of CS's that all devices must implement to claim "compliance." Generic Packet Convergence Sublayer (GPCS) solves these problems.
Purpose	The purpose of this contribution is to define a Generic Packet Convergence Sublayer for Supporting Multiple Protocols over 802.16 Air Interface.
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve (s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.

Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, provided the IEEE receives assurance from the patent holder or applicant with respect to patents essential for compliance with both mandatory and optional portions of the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:chair@wirelessman.org> as early as possible, in written or electronic form, if patented technology (or technology under patent application) might be incorporated into a draft standard being developed within the IEEE 802.16 Working Group. The Chair will disclose this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/notices>.

A Generic Packet Convergence Sublayer Supporting Multiple Protocols over 802.16 Air Interface

1. Background

As requested by 802.16g Task Group during the comments resolution discussions in session #39, we are submitting a follow-up contribution on our Generic Packet Convergence Sublayer (GPCS) proposal i.e., C802.16g-05/25, by presenting additional supporting materials and integrating suggestions and comments received from other members.

2. References

[802.16-2004] : IEEE-Std 802.16 – 2004

[16e/D12] : IEEE 802.16e/D12

[16cor1/D5] : IEEE 802.16-cor1/D5

[16g-05/008r1] : IEEE 802.16g-05/008r1

3. Problems with the Current 802.16 Packet Convergence Sublayer

This section describes some problems with the currently-defined packet convergence sublayers.

3.1.Not Upper-Level Protocol Agnostic

A problem with the 802.16 standard is that it tries to address upper layer protocol issues (see section 5.2 in 802.16-2004). For each upper layer protocol, it defines how fields in an upper layer protocol header are used to determine the 802.16 scheduling service type, and thus the 802.16 connection. And the standard goes further to define how upper layer headers should be encoded (e.g., compressed). Our contention is that such issues are best left outside the 802.16 standard.

The authors contend it is difficult for the industry to accept a set of 802.16 convergence sublayers that all devices must implement to be called “compliant.” For example, if Ethernet CS, why should a phone have to implement Ethernet frame formats? And if IPv4: why should all 802.16 devices need to participate in IP address assignment, IP mobility, tunneling, etc? And if a vendor implements a proprietary upper layer protocol, how can its 802.16 layer be tested to be compliant? This contribution suggests that a generic packet convergence sublayer can help by simplifying a “compliance profile” that is independent of the upper layer protocol. Other bodies such as the *WiMAX Forum* may benefit from this simplification.

By way of example, some other (successful) link-layer protocol standards which do not address upper layer protocol-header-mapping, encoding and compression are: 802.3, 802.2, 802.11, Frame Relay and packet-over-SONET (POS) embodied in *Generic Framing Procedures* of ITU-T G.7041/Y.1303. Also, some yet-to-be-successful link protocols follow the same convention of leaving the upper layer standards body to define its encapsulation: Infiniband, 802.17 (Resilient Packet Ring). For instance, IP datagram encoding and compression for specific link layers is typically left for the IETF to define.

A simple example to consider is IP over Ethernet, which is fairly well understood. The IEEE 802.3 and 802.2 standards do not define how IP packets are mapped to Ethernet links or how IP addresses are assigned or mapped to Ethernet destination addresses. Those functions are specifies by the community, using the well-known protocols ARP and DHCP. Ethernet has a simple way of transporting priority bits (P-bits) in each packet (see 802.1Q) and QoS-aware Ethernet switches use them to schedule packets in the presence of congestion. But 802.1Q does not specify how the IP layer uses the P-bits. Similarly, we do not think 802.16 needs to specify how IPv4 TOS and DSCP fields map to 802.16 scheduling service types (see section 802.16-2004, section 11.13.19.3). However, the authors of this contribution recognize that Ethernet-based QoS took a long time to “get right,” and recognize the efforts of 802.16 to “get it right the first time.”

Other example IETF RFCs that explain how IP uses a link layer protocol are RFC2464 *Transmission of IPv6 Packets over Ethernet Networks*, RFC2625 *IP and ARP over Fibre Channel,* RFC2516, *A method for transmitting PPP over Ethernet (PPPoE)*. Also refer to work currently under way in the IETF working groups, IP-over-Infiniband (ipoib) and IP-over-Resilient Packet Ring (iporpr).

3.2.No Standard Service API

The 802.16 standard not does specify a thorough MAC service API. Although the MAC SAP is present in Figure 1 of 802.16-2004 (copied below, Figure 1), the MAC service definition found in Annex C is "informative" rather than normative. Instead of a standard service interface definition, 802.16 "prefers" to define "convergence" with upper layer protocols. We think the reason the standard instead chose the convergence layer approach was to foster interoperability by explicitly stating how upper layers' QoS definitions must map to 802.16 scheduling and security services. While this was a noble goal, we think it turned out to be too complicated and the standard as it stands now does not define all the procedures necessary to build many types of interoperable systems.

A standardized service API would provide a clean way to expose 802.16 services to the next layer in a protocol stack, or to 802.16-aware applications. Citations of some other MAC-layer service APIs are: 802.3-2002 section 2, the "MSAP" concept in IEEE Std 802-2001 (Figure 1), IEEE Std 802.11, 1999, section 6.2.

The GPCS can provide access to a standard service API for the case when all classification functions is NULL. Note this is equivalent to what used to be defined as "No Convergence Sublayer" before it was removed in IEEE 802.16-cor1/D5. However, GPCS does not attempt to define a standard service API.

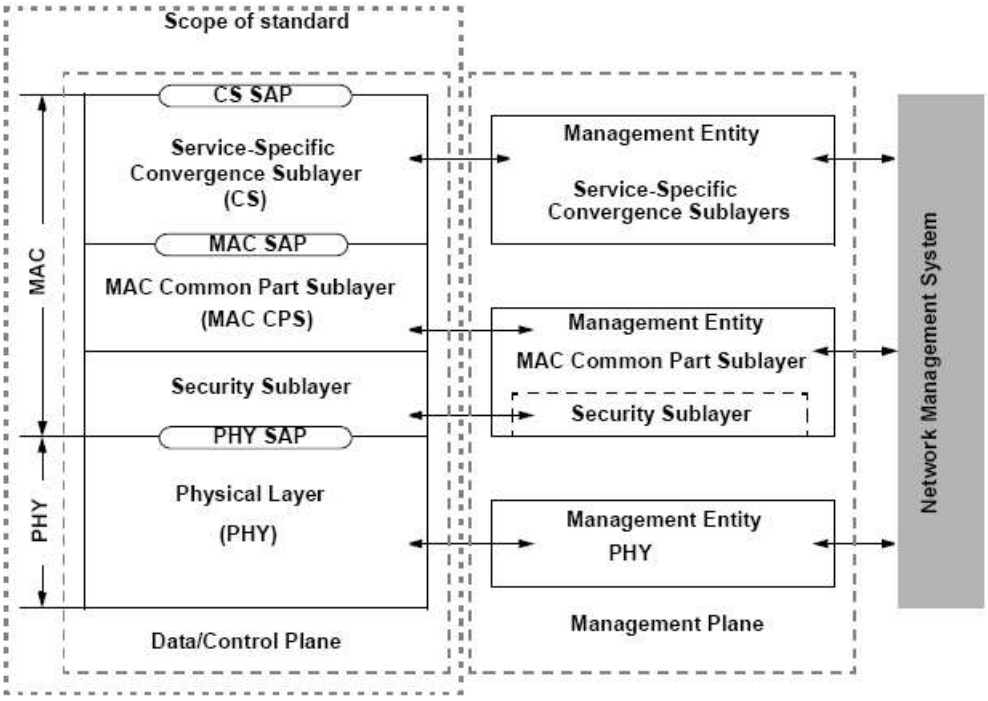


Figure 1—IEEE Std 802.16 protocol layering, showing SAPs

Figure 1: 802.16 Layering Model (copied from 802.16-2004)

3.3.Only One Upper Protocol per per Connection

The 802.16 convergence sublayers do not define the capability for multiple upper layer protocols to transport the SDUs on a single 802.16 connection. A connection ID (CID), is a valuable resource in both base stations and subscriber stations. An 802.16 system should not be forced to open a different CID for each upper layer protocol if packets for upper layer protocols have the same QoS requirements (802.16 scheduling service

types). The generic convergence sublayer provides a simple way to transport multiple protocols over a single 802.16 connection.

3.4.Poor Extendibility

It would appear that a convergence sublayer (CS) is needed for every type of protocol transported over the 802.16 MAC. Every time a new protocol type needs to be transported over the 802.16 air interface, the 802.16 standard needs to be modified to define a new CS type. A good example of the proliferation of CS's is found in Section 11.13.19.1 CS Specification. In 802.16e/D12, there are 4 new convergence sublayers that have been added in order to support IP or Ethernet/802.3 with IETF IETF header compression protocols ROHC and ECRTP. This is in addition to the 8 Convergence Sublayers already defined in 802.16 – 2004, that brings the total of Convergence Sublayers in 802.16e to 12!

802.16 should define an access method that allows for off-the-shelf components, those not even "aware" of 802.16, such as bridges, routers, NAT gateways, and classification engines to be used as components to build 802.16 compliant systems. For instance, bridges can operate just on 48-bit MAC addresses and logical port numbers, routers operate on IP addresses and logical port numbers, NAT gateways translate IP headers without knowledge of underlying link technologies, and classification subsystems can operate on a variety of upper-layer protocol packets to identify QoS flows.

Since the existing 802.16 convergence sublayers attempt to define upper layer protocol header interpretation, the 802.16 layer needs to change whenever upper layer systems are enhanced. This model is not extensible. 802.16 SAP layering should attempt to isolate itself from upper layers so upper layers can be enhanced in standardized or proprietary was and still operate effectively over 802.16.

3.5.Repeating Upper Layer Functions

In the current 802.16 specification, address-based classification rules define how data packets of different users are mapped to different CIDs, so that the differentiated QoS and/or security provisions can be provided. Particularly, it is critical for the downlink (DL) because the BS typically would use the address-based classification rules to map each packet to a different SS/MS in a point-to-multipoint topology.

The address-based classification rules require the current 802.16 convergence sublayer to maintain the mapping information between upper-protocol-defined addresses (e.g., IP or Ethernet) and CIDs. This ultimately forces the 802.16 CS to implement some upper layer functions.

For example, the IPv4 CS at BS maintains a "mapping state" for IP addresses to CIDs. Whenever there is a change in IP addresses, the mapping state needs to be updated. In non-802.16 systems and protocol stacks, upper layer address assignment and mapping to link layer entities is typically part of a routing function at the network layer.

A trouble with Ethernet headers is the size. So 802.16 has attempted to define a CS PDU format to deal with compress-header Ethernet PDUs (see section 5.2.7 of IEEE-802.16e/D12. Fields in the Ethernet header and additional fields inside the Ethernet payload (e.g., IP header and IP with mobile-IP header) complicate the classification process. Rather than pull-in Ethernet classification to 802.16, it would seem better to enable a system to classify an Ethernet packet before it is compressed. GPCS leaves the choice of classification method outside of 802.16.

Since upper layer protocol (ULP) addresses are used for some 802.16 packet CS classification rules, the packet CS layers must participate in upper layer protocol address-management procedures. When a ULP system changes addresses (mobility re-registration), translates addresses (NAT) or tunnels with nested headers (IPsec and Mobile IP), the 802.16 packet CS must be informed of changes to maintain consistent classification. The authors contend when layers "intertwine," it is difficult to write a specification that serves a wide enough variety of implementation and ULP layering options, and eventually industry interoperability will suffer.

Also, note that 802.1D bridging can be implemented over 802.16 without Ethernet convergence. For instance,

a BS can implement a bridge of VLANs between resilient packet ring and a set of 802.16 non-Ethernet-capable subscribers. 802.16 should not preclude the configuration of such topologies.

The authors of this contribution contend that although Ethernet Packet CS is great for certain classes of devices, interoperable "convergence" is difficult to define because there are so many implementation alternatives. So, Ethernet usage is best left to define "above" 802.16 so an Ethernet-based device can claim 802.16 compliance whether it implements "standard" Ethernet Packet CS or "proprietary" Ethernet CS. 802.16 should not define the use of Ethernet, or preclude flexibility in proprietary implementations.

3.6.Lack of Support for Multiple Header Compression Schemes

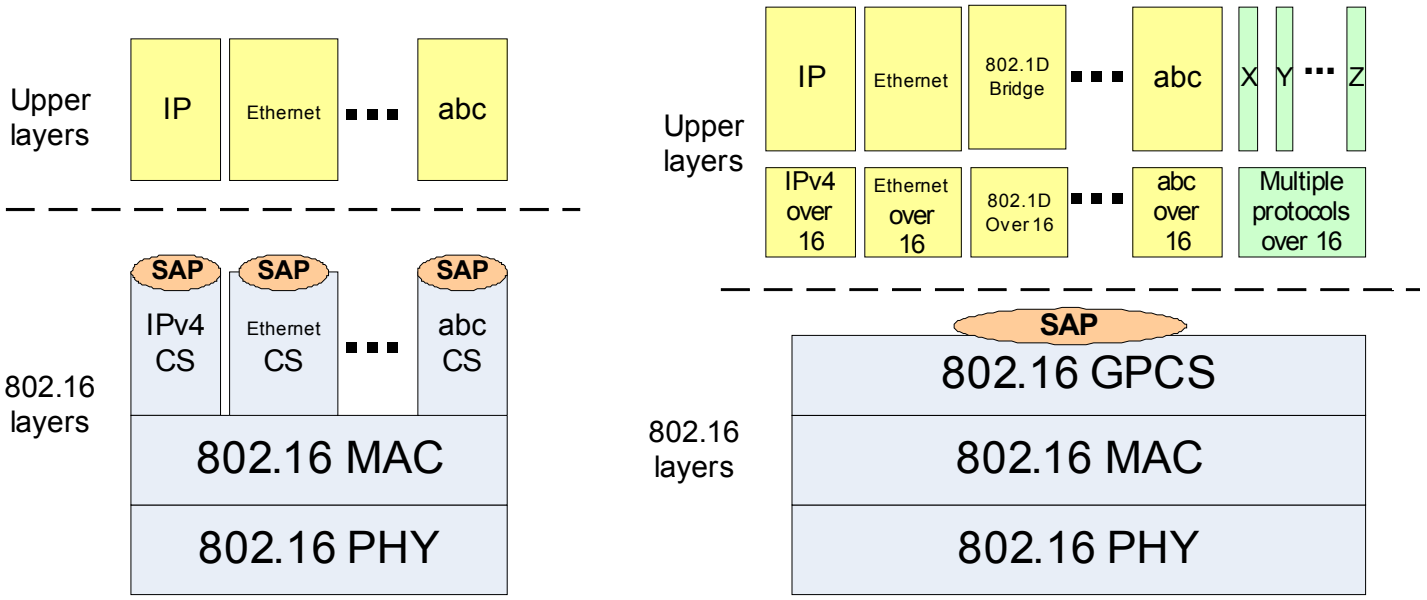
Furthermore, we see an architectural issue related to header compression protocols. For example, ROHC compresses all the information used by 802.16 to classify a packet to a CID and thus requiring that the implementation of ROHC (from standardization perspective at least) must be done after the 802.16 classification and thus within the 802.16 protocol stack. Note that 802.16e/D12 attempts to define classification methods for multiple header compression schemes. But it does not allow the use of proprietary header compression methods. The authors of this contribution claim that inclusion, and even mention, of header compression schemes, over-complicates the standard and leaves the standard in a state where new header compression schemes force revision of the standard. We think this will ultimately confuse the industry, and complicate compliance and interoperability tests. The 802.16 standard should allow for any header compression scheme to claim "compliance." Still, compatible and interoperable header compression needs to be defined, and classification methods need to be implemented. But the authors of this contribution contend the procedures do not need to be defined in the 802.16 standard.

4. Generic Packet Convergence Sublayer (GPCS)

To address problems with the current 802.16 packet convergence sublayer, cited in section 3, we propose a Generic Packet Convergence Sublayer (GPCS), defined as follows:

- GPCS provides a generic packet convergence layer. This layer uses the MAC SAP (see 802.16 Annex C) and exposes a new SAP to GPCS applications
- GPCS does not re-define or replace other convergence sublayers. Instead, it provides a SAP that is not protocol specific
- The basic function of the GPCS is still classification. It participates in the process to map upper layer packets (carried in 802.16 MAC SDUs) to appropriate 802.16 connections. But with GPCS, upper layer packet parsing (header inspection and interpretation to locate and extract fields relevant to classification rules) happens "above" GPCS. The results of packet parsing are classification parameters given to the GPCS SAP for "parameterized classification," but upper layer packet parsing is left to the GPCS application
- GPCS defines a set of SAP parameters as the result of upper layer packet parsing. These are passed from upper layer to the GPCS in addition to the data packet. Each is defined in later sections of this contribution
- ~~LOGICAL_PORT_ID~~
- ~~COS_ID (Class of Service ID)~~LOGICAL_FLOW_ID
- PROTOCOL_TYPE
- GPCS provides an optional way to multiplex multiple layer protocol types (e.g., IPv4, IPv6, Ethernet) over the same 802.16 connection. We call this MULTIPROTOCOL_ENABLE, a feature which can optionally be activated per CID. The capability of supporting this feature is indicated in a TLV parameter of the REG~~SBC~~ messages.
- For vendors to build interoperable equipment, each upper layer protocol type needs an interface

specification (e.g., IPv4 over 802.16, or Ethernet over 802.16). Such a standard specification can be developed by 802.16 or any other standard bodies, e.g., WiMAX, or IETF. Essentially, some of those specificaitons could be equivalent in function to protocol-specific packet convergence sublayers defined in today's 802.16 standard.



(a) Protocol structure with the service-specific Packet Convergence Sublayer

(b) Protocol structure with the proposed Generic Packet Convergence Sublayer (GPCS)

Figure 2: GPCS Layering Model

Figure 2 illustrates GPCS laying similarity to existing packet CS layers. Such a protocol structure of link layer interface to the upper layers has been widely used in some successful link layer protocol standards, e.g., 802.3, 802.2, 802.11, 802.17, Frame Relay, etc.

Comparing to the current 802.16 packet convergence sublayer, the proposed GPCS:

1. Decouples the 802.16 link layer from the higher layers protocols thus enabling the transport of new multiple upper layer protocol data over 802.16 air interface without requiring any modifications to 802.16 protocol
2. Allows the 802.16 CS to conduct the classification function without the need to interpret (parse) upper layer protocol headers, by allowing the higher layers protocols to pass some available information to the 802.16 CS
3. Avoids the repetitions of upper layer functions at the 802.16 CS
4. Allows multiple protocols over the same 802.16 MAC connection
5. Significantly simplifies the 802.16-comformance test of the packet convergence sublayer, by pushing the solutions to these problems to the upper layer. Although the generic convergence sublayer does not solve interoperability issues for Ethernet-bridging-oriented devices, or IPv4-oriented devices, GPCS provides opportunities for experts of upper protocols, e.g., experts from Ethernet community and experts from IP community such as IETF, to review and adopt standards for using 802.16 to transport IP with differentiated services. **Simple devices that have neither Ethernet nor IP, nor one of the dozen 802.16 convergence sublayers, can use 802.16 with GPCS and be called "compliant" with 802.16.**

A Generic Packet Convergence Sublayer (GPCS) would allow us to decouple the 802.16 link layer protocol from the higher layer protocols. In other words, instead of forcing 802.16 to know much about all the protocols it caries and possibly repeat some of the higher layers protocols functions within the 802.16 layer, the higher layers protocols need only to pass few parameters to enable 802.16 to classify the SDU.

To enable upper protocol independence, we propose two parts: parameters for the 802.16 SAP for the convergence sublayer, and an option per-SDU field prepended to each MAC SDU.

4.1. GPCS SAP ~~Parameters~~

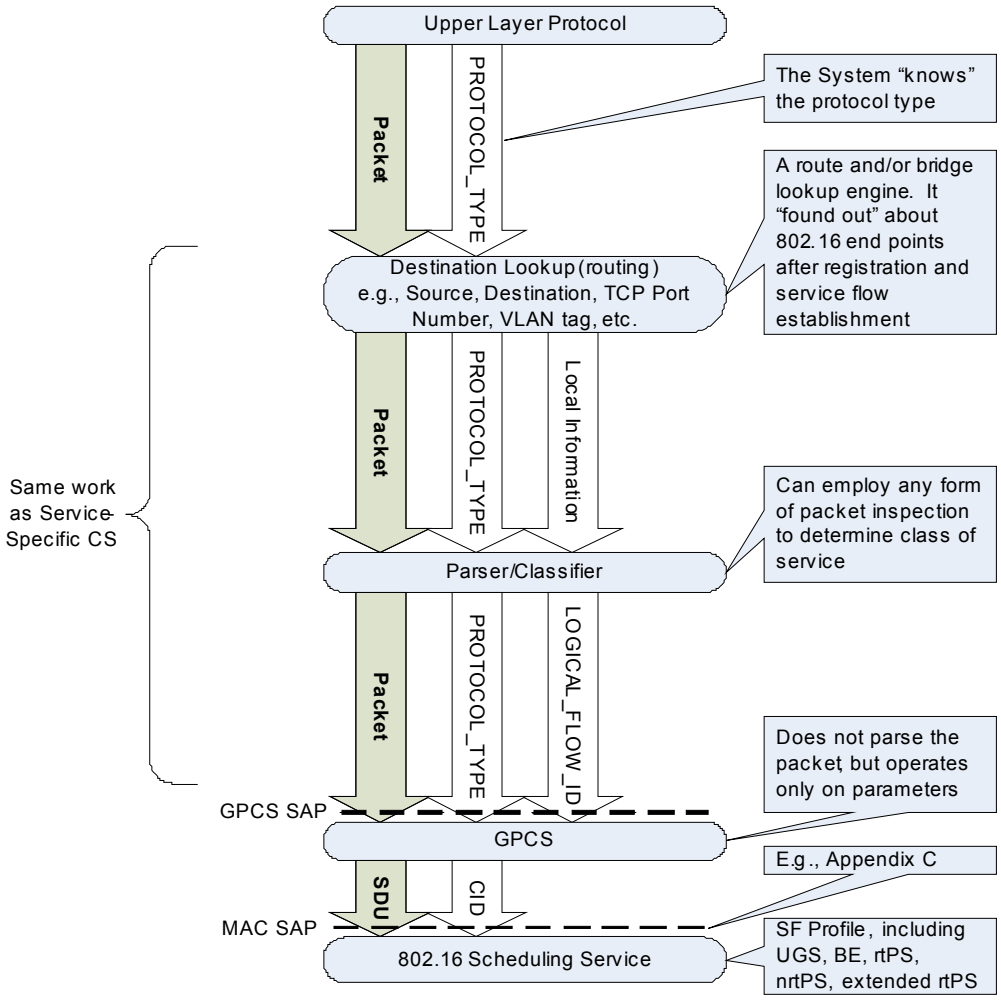
The GPCS SAP is embodied in two primitives, GPCS_DATA.request and GPCS_DATA.indication, which are defined in the normative section of this contribution. The proposed GPCS SAP parameters enable the upper layer protocol to generically specify 802.16 services without the need for the 802.16 layer to interpret upper layer protocol headers. In other words, since the SAP parameters are explicit, the parsing portion of the classification process is the responsibility of the upper layer. The SAP parameters need to be specified in the standard to provide an interface point where compliance can be measured independent of the upper layer protocol type (Ethernet, IPv4, etc.).

Figure 3 depicts the GPCS SAP parameters model. The figure intends to show how GPCS parameters are chosen in a system. It illustrates that ~~LOGICAL_PORT_ID and COS_ID abstractions can be implementation-specific, locally-defined constructs~~LOGICAL_FLOW_ID is a locally-define construct. We do not think that LOGICAL_~~PORT_ID and COS~~FLOW_ID need~~s~~ to have a code-point~~s~~ assigned and each code-point usage explained in a standard. The reason for such abstraction is to enable a wide variety of system configurations above the GPCS SAP that can be combined in various ways. Off-the-shelf components and subsystems can provide bridging, routing and parsing functions without adhering to a standard. The number of bits and numeric assignments ~~of those parameters are~~is left to the implementation. However, a GPCS implementation must directly map LOGICAL_FLOW_ID to exactly one 802.16 service flow and thereby, exactly one connection.

~~Since the GPCS SAP carries abstract parameters, GPCS SAP compliance does not require specific parameter values be standardized. Rather, c~~Compliance must be measured by the *results* of GPCS classification. For instance, the GPCS layer in a given vendor's system generates a CID with a certain security profile and 802.16 scheduling service type under known conditions, employing either standard packet-parsing convergence or a proprietary method. A compliance tester can then validate that the CID's security profile is operating correctly and the scheduling service is delivering the required throughput and latency. With GPCS, 802.6/GPCS compliance can be tested independently of specific upper layer classification methods. Without GPCS, 802.16 must define convergence and classification for each method that deserves interoperability and thus compliance. Considering the variety of possible upper layer protocol stack configurations, and the possibility of proprietary methods, it is difficult if not impossible for all "deserving" systems to claim 802.16 compliance without GPCS.

Note that Figure 3 does not show where header compression is performed. We believe that header compression is a facility that is orthogonal to classification and is best defined separately from the convergence process. For example, a system implementing IPv4 with ROHC could implement header compression in the "Upper Layer Protocol" or a compressor engine could be inserted just below the "Parser/Classifier." Either configuration could result in an ROHC-interoperable system.

Also note that a system could define a *null* function for "Destination Lookup," "Parser/Classifier" and even "Generic Classifier." The upper protocol, or "application" in such a system must be 802.16-aware and provide direct access to the MAC SAP. For instance, a video-over-802.16 device need not define Ethernet or IP convergence function. Other standards could build on 802.16 GPCS, and proprietary systems could be defined. Such systems could be called "Classification Free." The authors think it would be great if they all could achieve 802.16 compliance without requiring modifications to the 802.16 standard.


Upper Layer Protocol
Packet
PROTOCOL_TYPE
The System "knows" the protocol type
A route and/or bridge lookup engine. It "found out" about 802.16 end points after registration and service flow establishment
Destination Lookup (routing) e.g., Source, Destination, TCP Port Number, VLAN tag, etc.
Packet
PROTOCOL_TYPE
Local Information
Same work as Service-Specific CS
Can employ any form of packet inspection to determine class of service
Parser/Classifier
Packet
PROTOCOL_TYPE
LOGICAL_FLOW_ID
Does not parse the packet, but operates only on parameters
GPCS SAP
GPCS
E.g., Appendix C
SDU
CID
MAC SAP
SF Profile, including UGS, BE, rtPS, nrtPS, extended rtPS
802.16 Scheduling Service

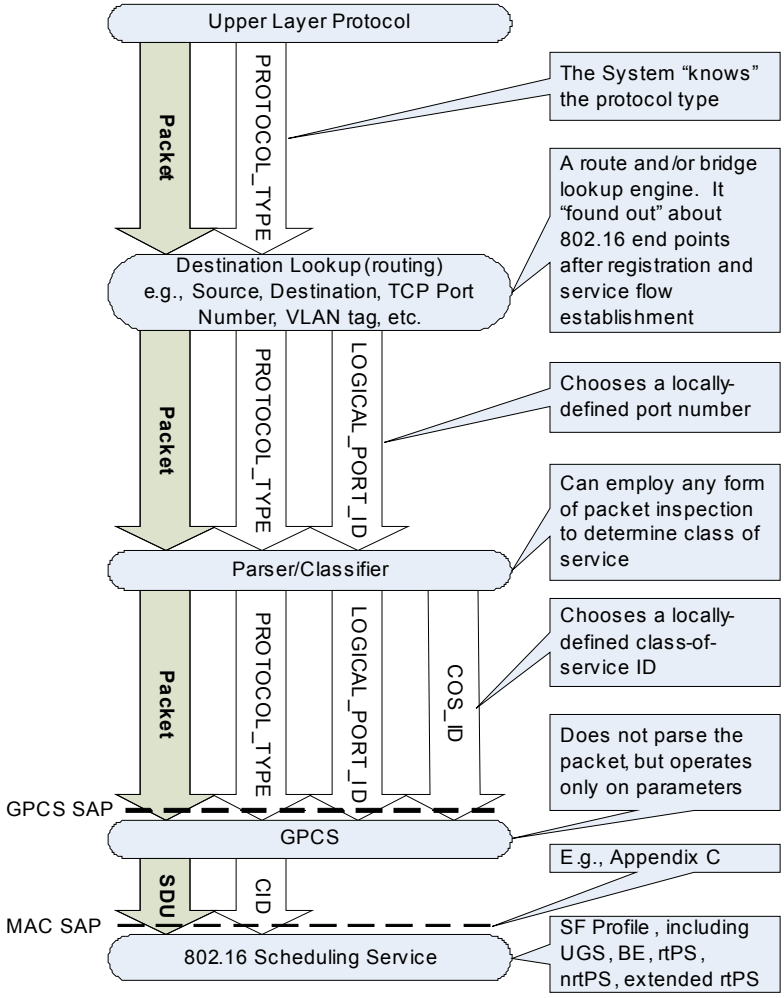


Figure 3: GPCS SAP Parameters Model

~~GPCS defines SAP parameters abstractly because the authors think their use in a local system is outside the scope of the 802.16 standard. For instance, bit width, code points, error checking, programming language and operating system bindings are not specified. But PROTOCOL_ID, which since it can optionally be transmitted between 802.16 end points, needs to have code points assigned (such as Ethernet, PPPoE, IPv4, IPv4-ROHC, IPv6, etc.).~~

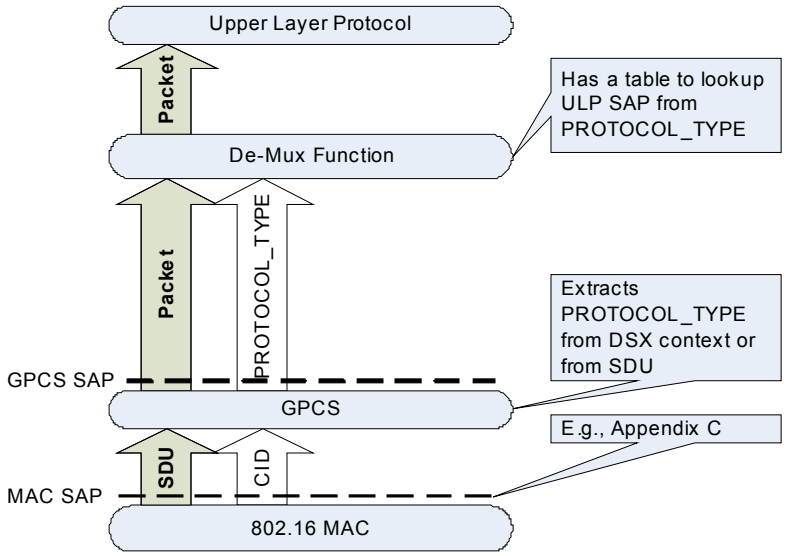


Figure 4: PROTOCOL_TYPE Usage by Receiver

SAP Parameters:

PROTOCOL_TYPE. The protocol type field identifies the upper layer protocol that is immediately above the 802.16 protocols. If MULTI_PROTOCOL_ENABLE is activated, the PROTOCOL_TYPE shall be also carried over-the-air in every SDU, so the 802.16 receiver can demultiplex an SDU and invoke the appropriate upper layer protocol. If MULTIPROTOCOL_ENABLE is not activated, an 802.16 CID can carry only one protocol type, and the PROTOCOL_TYPE field is only communicated during connection establishment. It is thus analogous to the "ethertype" field in an Ethernet packet (`http://www.iana.org/-assignments/ethernet-numbers`) or the 16-bit PPP data link (DL) layer protocol field in PPP packets (`http://www.iana.org/assignments/ppp-numbers`)[1]. **The authors recommend that 802.16 start a new IANA registry of GPCS protocol numbers**. Note that the Ethernet numbers probably should not be used because it has no Ethernet code point which could mean Ethernet-over-802.16. PPP protocol numbers aren't good either because although they include Ethernet, ROHC, IP, etc., it does not include a protocol number for PPP.

Note that GPCS with MULTIPROTOCOL_ENABLE is just about like Ethernet Packet CS except no Ethernet addresses are transported.

LOGICAL_FLOW_ID. This parameter is a locally-assigned number, unique within a GPCS implementation. A GPCS implementation must have the capability to map this parameter directly to a service flow ID (SFID). A GPCS upper layer is expected to establish an 802.16 service flow using the Add_Service_Flow.response (ASF.response) (see section 14.5.6.4.4.2). The upper layer uses the SFID in the ASF.response to chose the appropriate LOGICAL_FLOW_ID.

~~LOGICAL_PORT_ID. A port number that is output from a bridging or routing function. It is locally assigned and not exchanged between 802.16 end-points. For instance, an 802.1D bridge agent in a BS could discover a connection to a specific SS and assign a LOGICAL_PORT_ID number to that connection plus SS entity. The LOGICAL_PORT_ID can thus become a destination link for an 802.1D bridge. It could transact spanning tree and GARP messages and relay and filter packets based on 48-bit 802 MAC addresses and VLAN tags. The exact mechanism for how this is done is left to the implementation. In 802.1D terminology, the LOGICAL_PORT_ID could be called a "Bridge Port ID," which a locally-assigned number. Note that the process for connection establishment on the air interface is defined in 802.16, but apart from Annex C, the end-point's process to communicate service flows and~~

[1] Note the wide variety of header-compression protocol types defined for PPP. Since PPP is intended for bandwidth-constrained links, it seems to have very similar header compression requirements as 802.16. Also note the set of code points defined for various bridging functions.

~~connections is not. An 802.1D compliant bridge can implement SS and connection discovery however it needs to, and 802.1D bridging over 802.16 can still be standardized and interoperable.~~

~~COS_ID (Class of Service ID). A class of service identifier output from a packet-parsing classification function. The class of service could simply identify an 802.16 scheduling service type. Alternatively, a system could define it as a superset of 802.16 scheduling services, and perform local scheduling and/or queuing based on the value. So, instead of specifying COS_ID to mean exactly the 802.16 scheduling service type, it is left as an abstract locally-defined identifier. In either case, an implementation of the "Generic Classifier" takes the COS_ID as input and uses it to determine a CID with matching 802.16 scheduling services.~~

MULTI_PROTOCOL_ENABLE. This parameter is not shown in Figure 3. A local system can optionally choose to activate this feature to enable multiple PROTOCOL_TYPEs to be carried over a single CID. MULTI_PROTOCOL_ENABLE is useful for devices that are constrained to implement a small number of CIDs yet need to carry a variety of protocols using the same 802.16 scheduling service type. For instance, a single best-effort (BE) connection could carry statistically multiplexed packets of IPv6, IPv6-ROHC and Ethernet PROTOCOL_TYPEs.

SDU and Length. GPCS has SDU and SDU-length parameters~~, which are the same as the corresponding parameters in Annex C, MAC_DATA.request and MAC_DATA.indication~~.

5. Application Examples of the Proposed Generic Packet CS (GPCS)

This section describes some examples of how GPCS is used, particular for two upper protocols that have lots of interest: Ethernet packets, IPv4, and ROHC header compression.

5.1.IPv4-ROHC over GPCS

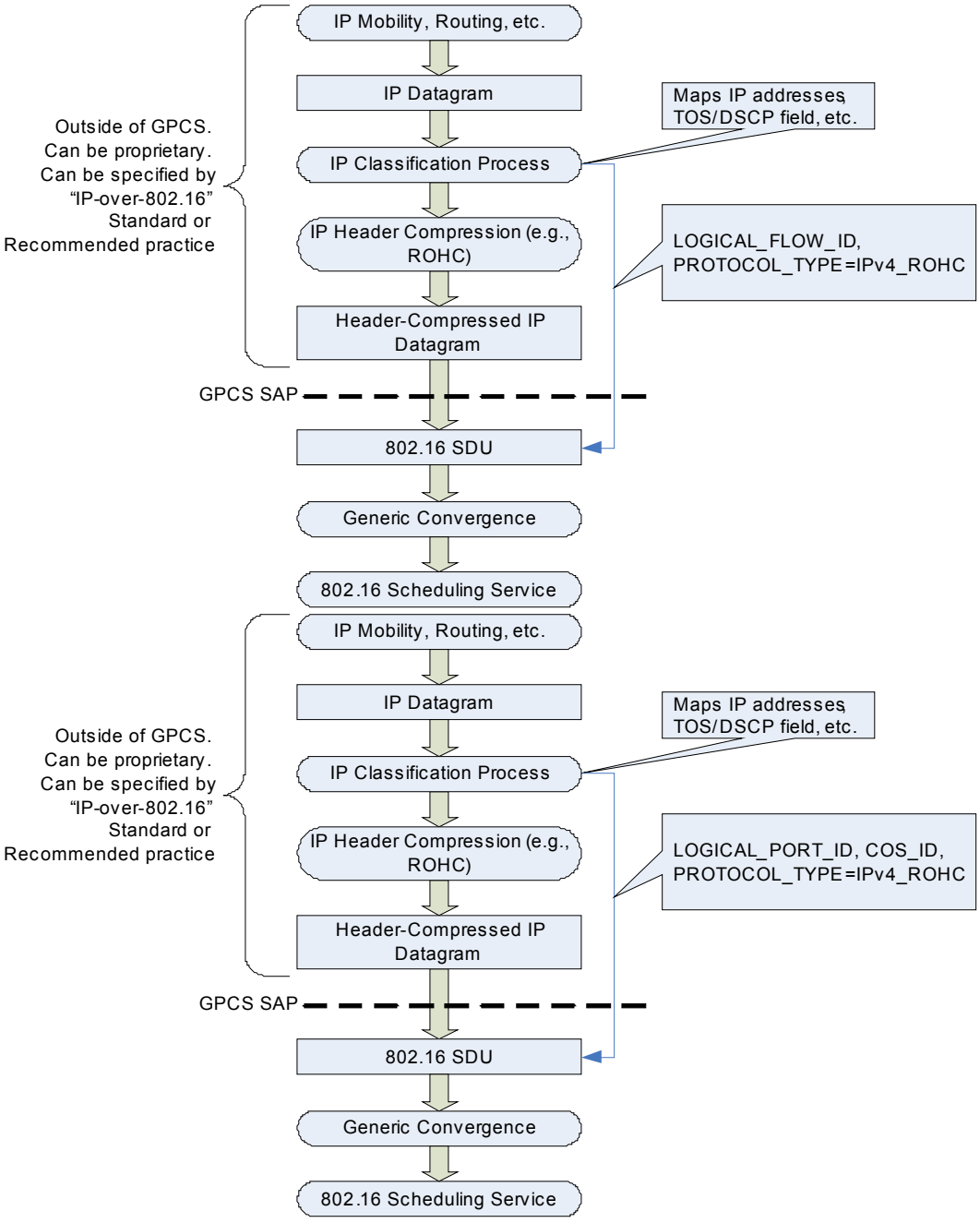


Figure 5: IPv4-ROHC over GPCS

Figure 5 illustrates an example local 802.16 "stack" transmit process.

An IP endpoint or router typically needs to map a destination IP address to an egress link and destination MAC address. For Ethernet, many systems have an IP route lookup process and address resolution (ARP) process. Same for 802.16. An 802.16 BS can implement an IP route process, along with SS discovery and ARP if necessary. Whatever the process, standards-based or proprietary, the output of route and address resolution is a path to a network entity. This path is included in a LOGICAL_~~PORT~~FLOW_ID, which identifies a local path from the IP entity to the egress radio for a subscriber station. Note that an implementation can employ a separate route engine to help (even a separate, off-the-shelf, non-802.16-aware IP router subsystem), or can combine IP packet classification (parsing and inspection) with routing.

The parsing and inspection process in a typical implementation results in more information to embody in the

LOGICAL_FLOW_ID: class of service~~a COS_ID~~. It may parse single-IP headers, tunneled IP-in-IP headers (e.g., for IP mobility), and even combine network address translation functions. As with routing, standard off-the-shelf subsystems may be employed to choose an appropriate, locally-defined ~~COS_ID~~LOGICAL_FLOW_ID.

Header compression can be performed anywhere above the GPCS SAP. In the figure, it is shown last, which may make sense for some systems that translate headers during the packet inspection process. Robust Header Compression (ROHC) (see IETF RFC 3095) can use GPCS. It should have its own protocol type, such as IPv4-ROHC so a receiver can "know" to apply the ROHC algorithm. Other header compression techniques can also be used, such as the "original" Van Jacobsen TCP/IP header compression," RFC1144.

If MULTIPROTOCOL_ENABLE is activated, GPCS requires the transmitter to insert~~s~~ its protocol type (IPv4) as an extended subheader entry so the receiver can de-multiplex: choose the right upper protocol entity (receive SAP).

~~The Generic Classifier implements a locally-defined function to map PROTOCOL_TYPE (IPv4-ROHC), LOGICAL_PORT_ID, and COS_ID to an appropriate CID. For example, LOGICAL_PORT_ID could be an index to a table in which each entry has a pointer a subscriber station's table of available service classes. COS_ID could be service-class index to a table of CIDs for the subscriber.~~

5.2.Ethernet over GPCS

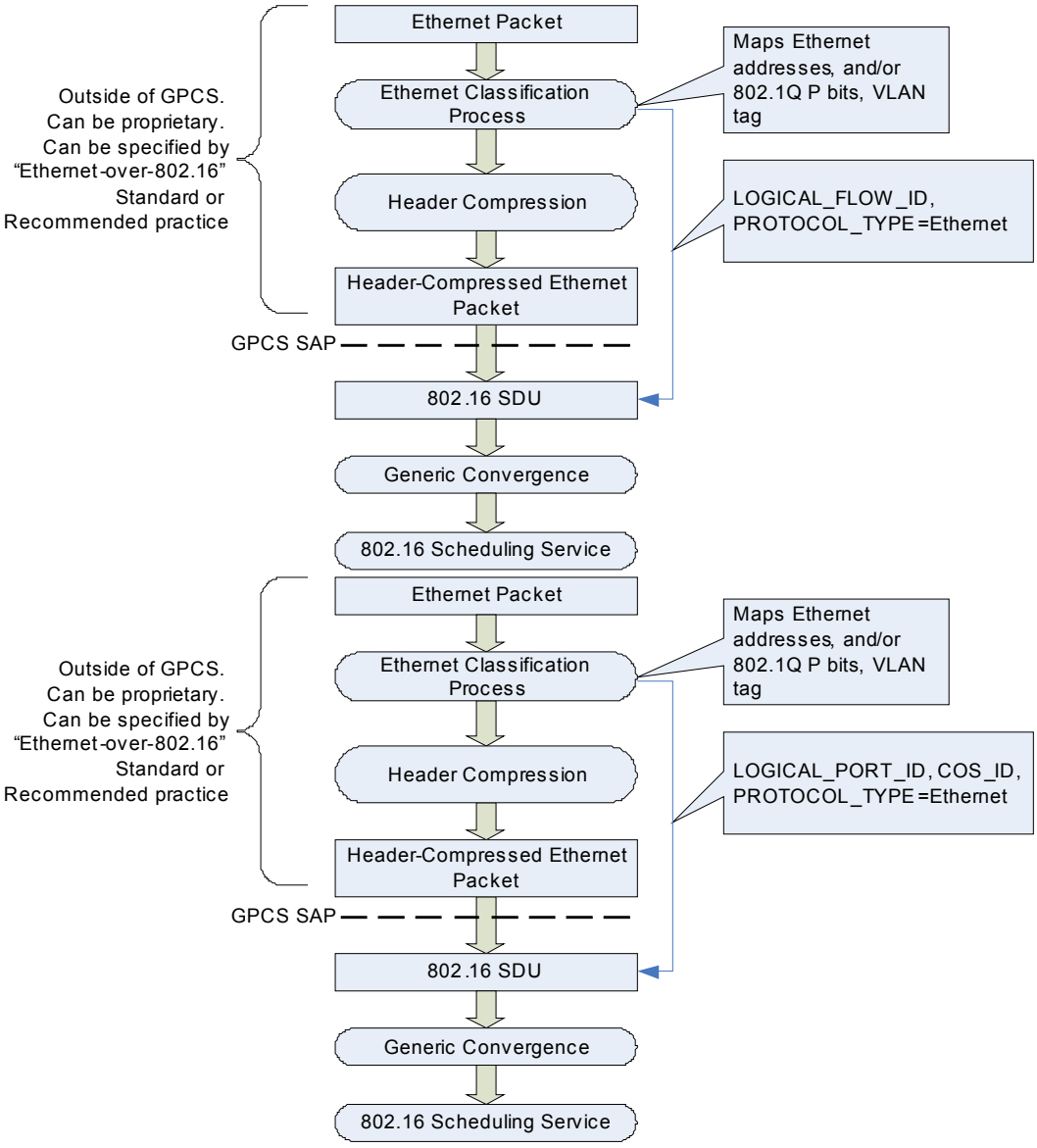


Figure 6: Ethernet over GPCS

5.3.802.1D,Q Bridging over GPCS

Figure 7 is copied from IEEE 802.1Q-2003, which shows 802.16 participating in an 802.1 bridged-LAN environment. The authors of this contribution contend that the 802.16 MAC should support 802.1 bridging in a "generic" way. By generic, we mean 802.16 should not require a system to emulate another MAC and/or frame formats of other MACs, such as 802.3, but should offer "service" to the 802.1 bridging protocols (802.1D, 802.1Q).

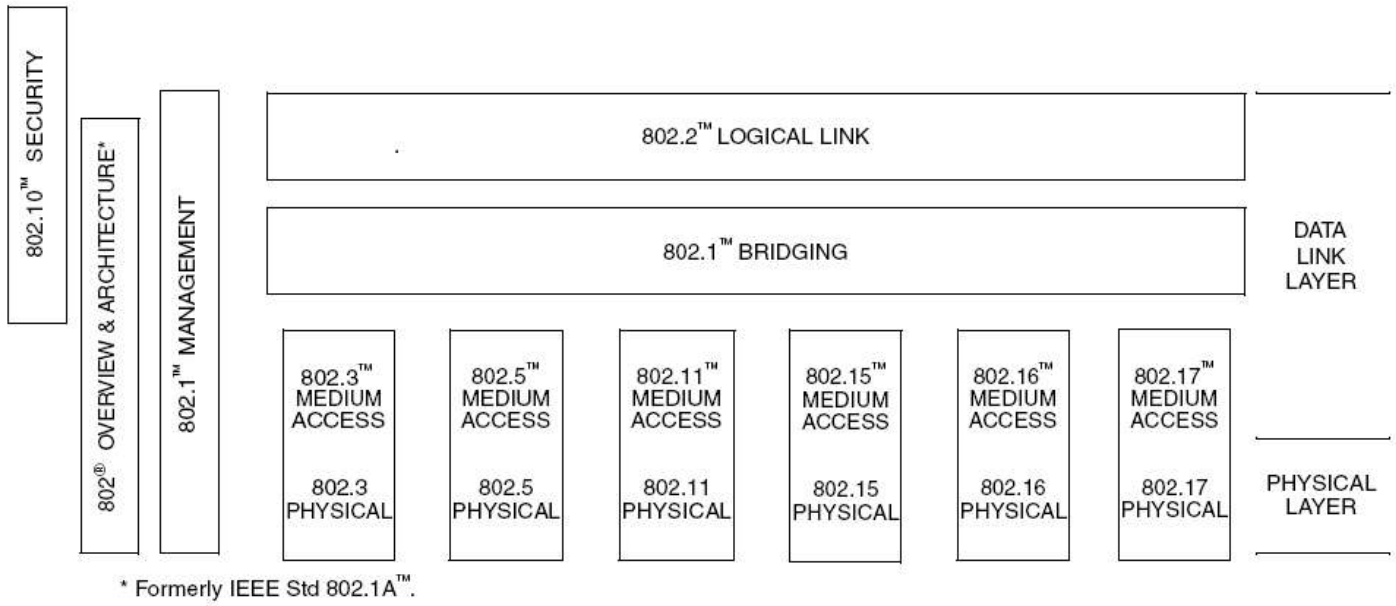


Figure 7: 802.1 Bridging -- Standards Relationships

However, GPCS by itself does not specify how bridge protocols are integrated with 802.16 particularly with respect to service flow and connection establishment and discovery. **The contributors had a desire to propose a "baby step" to simplify layering, but not a desire to solve the bridging issues**. Even so, the 802.16g specification could use GPCS to further define how 802.1 bridging uses GPCS. Figure 8 shows an example 802.1D LAN with 4 nodes. Following are some issues to solve:

1. GPCS does not specify all service access point (SAP) primitives for LAN-specific service flow or connection establishment
2. GPCS does not recommend how a bridge uses 802.16 broadcast and multicast connections to relay bridged packets:
    a. Multicast connection discovery or establishment. The bridging entity needs to be informed of connections. Regarding GPCS, it should only "care" about MAC address~~, LOGICAL_PORT_ID and COS_ID~~ and LOGICAL_FLOW_ID
    b. Map VLAN IDs to CIDs that participate in a corresponding 802.1Q multicast group (if multicast is desired)
    c. Map 802.1Q priority to ~~GPCS COS_ID~~appropriate LOGICAL_FLOW_ID
    d. 802.1 endpoint discovery. An 802.1D bridge makes its relay decisions based on 48-bit MAC addresses (typically uses the destination MAC address). When a bridge has a packet destined for and endpoint node (identified by destination MAC address), but does not "know" which port will reach the endpoint, the bridge can employ broadcast or multicast or other procedures particular to the MAC layer.
3. Define SAP primitives to expose 802.16 SS/MS MAC addresses to the bridge relay. For instance, an 802.16 MS could be an endpoint (not a bridge). The bridge function on the BS-side (or ASN) needs to "know" the mapping of the endpoint's MAC address to a logical link number, and therefore to a connection. Further, a VLAN-capable bridge needs to maintain a mapping of the 802.16 MAC address to VLAN IDs.

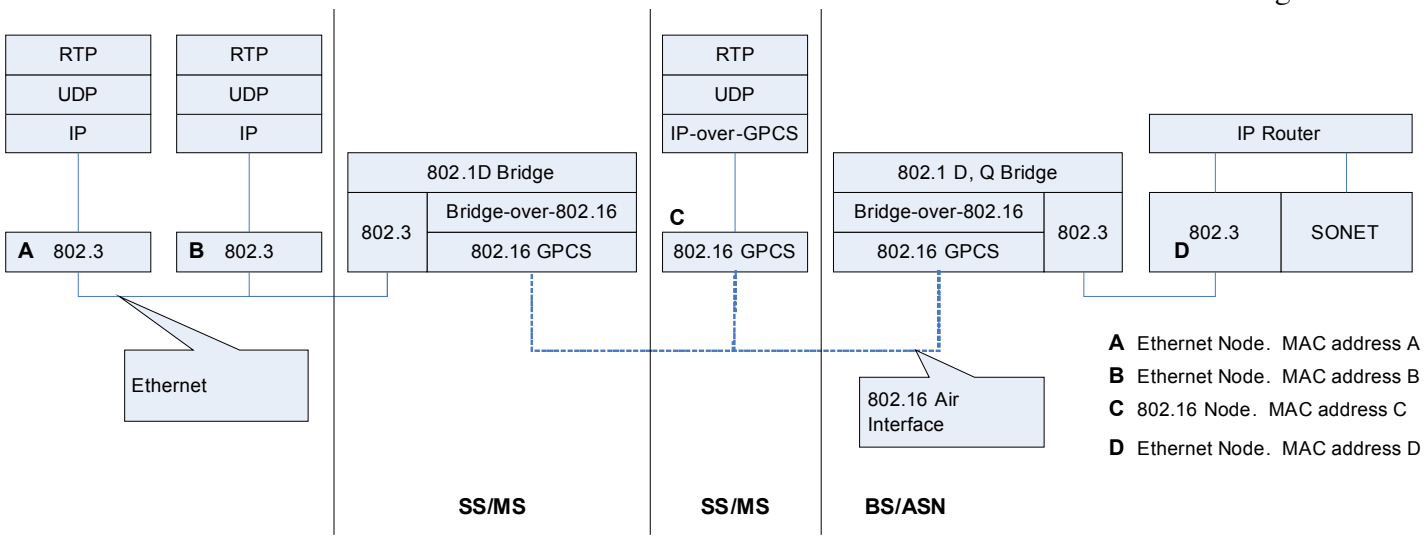


Figure 8: Example LAN with 4 endpoint nodes: A, B, C, D

6. Suggested Changes

In 802.16g-05/008r1, we propose the following changes (new text in blue):

1. page 3 replace line 1 to line 52 by the following text:

5. Service-Specific CS

5.2 Packet CS

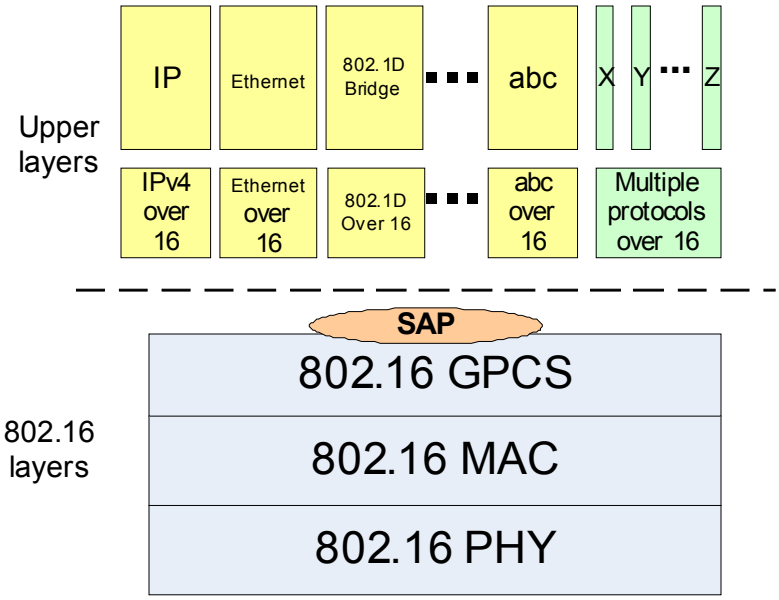
[insert new subclause 5.2.8 and subsequent test below]

5.2.8 Generic Packet Convergence Sublayer (GPCS)

The Generic Packet CS (GPCS) is a upper layer protocol-independent packet convergence sublayer that supports multiple protocols over 802.16 air interface. It is defined as follows:

- GPCS provides a generic packet convergence layer. This layer uses the MAC SAP and exposes a SAP to GPCS applications
- GPCS does not re-define or replace other convergence sublayers. Instead, it provides a SAP that is not protocol specific
- ~~The basic function of the GPCS is still classification. It participates in the process to map upper layer packets (carried in 802.16 MAC SDUs) to appropriate 802.16 connections. But w~~With GPCS, packet parsing happens "above" GPCS. The results of packet parsing are classification parameters given to the GPCS SAP for "parameterized classification," but upper layer packet parsing is left to the GPCS application
- GPCS defines a set of SAP parameters as the result of upper layer packet parsing. These are passed from upper layer to the GPCS in addition to the data packet. Each is defined in section 5.2.8.1.
- ~~LOGICAL_PORT_ID~~
- ~~COS_ID (Class of Service ID)~~LOGICAL_FLOW_ID
- PROTOCOL_TYPE
- GPCS provides an optional way to multiplex multiple layer protocol types (e.g., IPv4, IPv6, Ethernet) over the same 802.16 connection. A TLV parameter, MULTIPROTOCOL_ENABLE, is defined in the DSx messages to enable/disable this feature. . The capability of supporting this feature is indicated in a TLV parameter of the ~~SBC~~REG messages.

- For interoperability, each upper layer protocol type needs an interface specification (e.g., IPv4 over 802.16, or Ethernet over 802.16). Such a standard specification is out of scope of the GPCS.



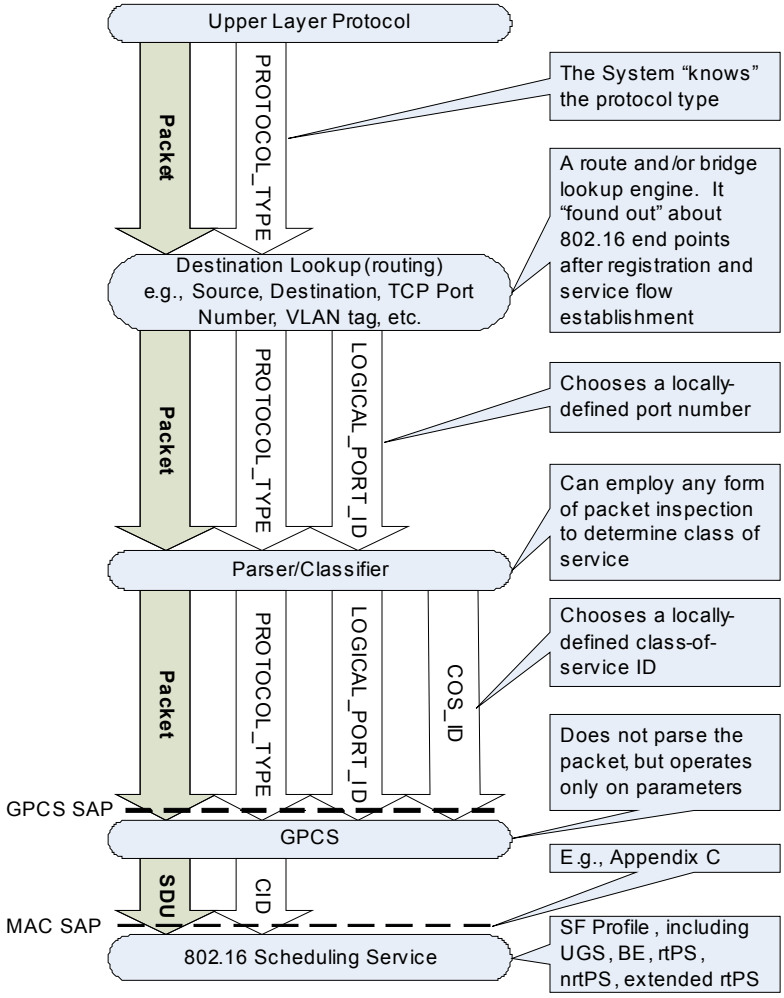
(b) Protocol structure with the proposed Generic Packet Convergence Sublayer (GPCS)

Figure 17c: GPCS Layering Model

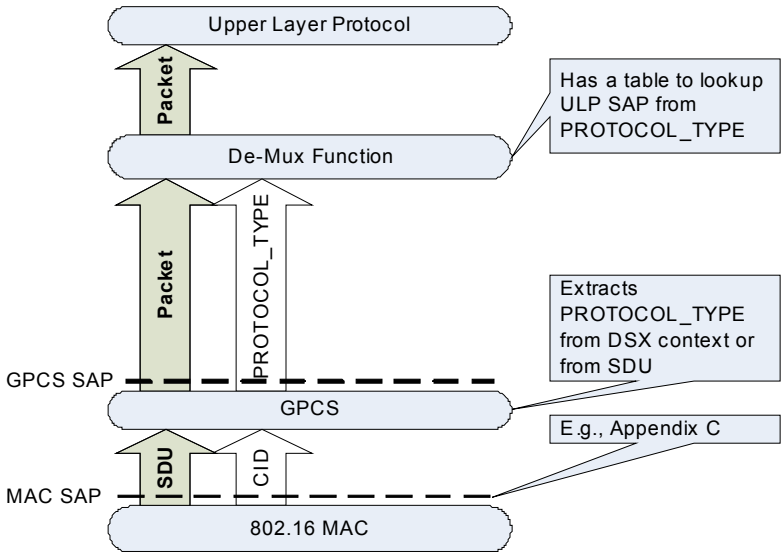
5.2.8.1 GPCS Service Access Point (SAP) Parameters

The GPCS SAP parameters enable the upper layer protocols to generically pass information to the GPCS so that the GPCS does not need to interpret upper layer protocol headers in order to mapping the upper layer data packets into proper 802.16 MAC connections. Since the SAP parameters are explicit, the parsing portion of the classification process is the responsibility of the upper layer. The parameters are relevant for SAP data path primitives, GPCS_DATA.request and GPCS_DATA.indication as described in sections 5.2.8.1.2 and 5.2.8.1.3, respectively. ~~[Note to editor: copy specification method from Annex C to this section to describe GPCS_DATA primitives and SAP parameters.]~~

~~There are four GPCS SAP parameters: LOGICAL_PORT_ID, COS_ID, PROTOCOL_TYPE, and MULTI_PROTOCOL_ENABLE. Figure 17d depicts the GPCS SAP parameters model. It shows how the first three GPCS parameters, LOGICAL_PORT_ID, COS_ID, and PROTOCOL_TYPE, can be chosen in a system. The parameter, MULTI_PROTOCOL_ENABLE, is required for signaling whether or not the multiple protocol packets are transmitted on the same CID.~~



~~Figure 17d: GPCS SAP Parameters Model~~



~~Figure 9: PROTOCOL_TYPE Usage by a Receiver~~

PROTOCOL_TYPE. The protocol type field identifies the upper layer protocol that is immediately above the 802.16 protocols. The PROTOCOL_TYPE is included in GPCS_DATA.request and GPCS_DATA.indication as defined in sections 5.2.8.1.2 and 5.2.8.1.3 respectively. If

MULTI_PROTOCOL_ENABLE is activated, the PROTOCOL_TYPE shall be also carried over-the-air in every SDU, so the 802.16 receiver can demultiplex an SDU and invoke the appropriate upper layer protocol. If MULTIPROTOCOL_ENABLE is not activated, an 802.16 ~~CID~~ connection can carry only one protocol type, and the PROTOCOL_TYPE field shall be included in ASF.indication primitive and DSx messages during connection establishment. ~~is only communicated during connection establishment through DSx messages. It is thus analogous to the "ethertype" field in an Ethernet packet or the 16-bit PPP data link (DL) layer protocol field in PPP packets.~~The PROTOCOL_TYPE values are the same as the value of the GPCS PROTOCOL_TYPE encoding as defined in section 11.13.19.3.4.3. ~~The GPCS can either uses the IANA to assign 2-byte 802.16 PROTOCOL_TYPE numbers, or maybe just borrow the number space from PPP. Note that GPCS with MULTIPROTOCOL_ENABLE is just about like Ethernet Packet CS except no Ethernet addresses are transported, and so they don't need to be header-compressed.~~

~~LOGICAL_PORT_ID. A port number that is typically output from a bridging or routing function. It is locally assigned and not exchanged between 802.16 air link..~~

~~COS_ID (Class of Service ID). A class of service identifier output from a packet-parsing classification function. The class of service could simply identify an 802.16 scheduling service type. Alternatively, a system could define it as a superset of 802.16 scheduling services, and perform local scheduling and/or queuing based on the value. So, instead of specifying COS_ID to mean exactly the 802.16 scheduling service type, it is left as an abstract locally-defined identifier. In either case, the GPCS takes the COS_ID as input and uses it to determine a CID with matching 802.16 scheduling services.~~

LOGICAL_FLOW_ID. This parameter is a locally-assigned number, unique within a GPCS implementation. A GPCS implementation shall map this parameter directly to a service flow. During service flow establishment, the 802.16 control plane function shall allocate a LOGICAL_FLOW_ID and return it to the upper layer in the Add_Service_Flow.response as defined in section 14.5.6.4.4.2.

MULTI_PROTOCOL_ENABLE. This parameter is not shown in Figure 17d. A local system can optionally choose to enable/disable multiple PROTOCOL_TYPEs to be carried over a single CID. MULTI_PROTOCOL_ENABLE is useful for devices that are constrained to implement a small number of CIDs yet need to carry a variety of protocols using the same 802.16 scheduling service type. For instance, a single best-effort (BE) connection could carry statistically multiplexed packets of IPv6, IPv6-ROHC and Ethernet PROTOCOL_TYPEs.

DATA. SDU data reference to a number of bytes delivered by the GPCS upper layer to GPCS, or by GPCS to the upper layer.

LENGTH. Number of bytes in DATA.

5.2.8.1.2 GPCS_DATA.request

5.2.8.1.2.1 Function

This primitive defines the transfer of data from the upper-layer, GPCS application, to the GPCS SAP.

5.2.8.1.2.2 Semantics of the Service Primitive

The parameters of the primitive are as follows:

```
GPCS_DATA.request
(
PROTOCOL_TYPE,
LOGICAL_FLOW_ID,
length,
data
)
```

The PROTOCOL_TYPE specifies the protocol function of the upper layer, as defined in section 11.13.19.3.4.3.

LOGICAL_FLOW_ID is a locally-assigned number, unique within a GPCS implementation. A GPCS implementation shall map this parameter directly to a service flow. During service flow establishment, the 802.16 control plane function shall allocate a LOGICAL_FLOW_ID and return it to the upper layer in the Add_Service_Flow.response as defined in section 14.5.6.4.4.2. Furthermore, a LOGICAL_FLOW_ID is mapped to exactly one 802.16 service flow identified by an SFID.

The length parameter specifies the length of the SDU in bytes. Note if MULTIPROTOCOL_ENABLE is active, the length does not include the PROTOCOL_TYPE which GPCS will prepend to the SDU before sending to the MAC SAP.

The data parameter specifies the SDU as received by the GPCS SAP from the upper layer protocol.

5.2.8.1.2.3 When Generated

This primitive is generated by an upper layer protocol when a GPCS SDU is to be transferred to a peer entity or entities.

5.2.8.1.2.4 Effect of Receipt

The receipt of this primitive causes GPCS to map the LOGICAL_FLOW_ID to a service flow and thereby a connection. GPCS invokes MAC functions, for example the MAC SAP (an example MAC SAP definition is provided in Annex C) to effect transfer of the SDU to the MAC layer. If MULTI-PROTOCOL_ENABLE is activated as defined in sections 11.8.13 and 11.13.19.3.4.19, then GPCS will insert the PROTOCOL_TYPE at byte number 0 of the SDU prior to delivering the SDU to the MAC layer.

5.2.8.1.3 GPCS_DATA.indication

5.2.8.1.3.1 Function

This primitive defines the transfer of data from the GPCS to an upper layer protocol.

5.2.8.1.3.2 Semantics of the Service Primitive

The parameters of the primitive are as follows:

```
GPCS_DATA.indication
(
PROTOCOL_TYPE,
length,
data
)
```

The PROTOCOL_TYPE identifies the specific upper layer protocol, as defined in section 11.13.19.3.4.-3. GPCS formulates the PROTOCOL_TYPE in one of 2 ways:

1. If MULTIPROTOCOL_ENABLE is activated as defined in sections 11.8.13 and 11.13.19.3.4.19, then GPCS locates PROTOCOL_TYPE starting at byte number 0 in the SDU.
2. If MULTIPROTOCOL_ENABLE is not activated (the default condition), GPCS obtains PROTOCOL_TYPE from the CID context since it was established in a DSx message when the corresponding 802.16 service flow was established.

The length parameter specifies the length of the SDU in bytes. Note if MULTIPROTOCOL_ENABLE is active, the length does not include the PROTOCOL_TYPE which GPCS will remove from the SDU

before sending to the upper layer.

The data parameter specifies the SDU as received by the GPCS SAP from the MAC.

Note that the GPCS_DATA.indication does not have a LOGICAL_FLOW_ID parameter because LOGICAL_FLOW_ID is not transferred over the 802.16 air interface.

5.2.8.1.3.3 When Generated

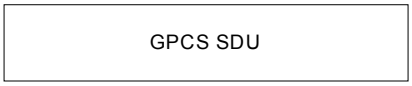
This primitive is generated by GPCS whenever a GPCS SDU is to be delivered to an upper layer protocol resulting from receipt of a MAC SDU by GPCS.

5.2.8.1.3.4 Effect of Receipt

The effect of receipt of this primitive by the upper layer protocol entity is dependent on the validity and content of the SDU. The choice of upper layer protocol entity is identified by PROTOCOL_TYPE.

5.2.8.2. GPCS PDU Format

There are two different formats of the GPCS PDU depending on the parameter value of MULTIPROTOCOL_ENABLE, as shown in Figure 17e. PROTOCOL_TYPE indicates the outermost protocol of the SDU. It is 16-bit number assigned from a set of possible values of the PPP data link (DL) layer protocol numbers. This space of numbers is maintained by the IANA. It shall be communicated to the MAC receiver in every SDU if MULTIPROTOCOL_ENABLE is enabled for a CID.



(a) **GPCS PDU format with MULTIPROTOCOL_ENABLE Disabled**

(b) **GPCS PDU format with MULTIPROTOCOL_ENABLE Enabled**

Figure 17e: GPCS PDU formats Sent to the MAC SAP from the GPCS

2. Page 35, line 5645, insert new section 11.78.413:

11.7.4 8.13 Multiprotocol over GPCS Support

This basic capability parameter indicates whether or not the multiprotocol over GPCS is supported. If multiprotocol is supported, a BS or SS may send MULTIPROTOCOL_ENABLE (section 11.13.19.3.4.19) in a DSx message.

Type	Length	Value	Scope
4529	1	Bit#0: 0=multiprotocol not supported (default) 1=supported Bit#1 to Bit#7: reserved	SBCREG-REQ, SBCREG-REP

3. page 5, line 49, replace from page 5 line 40 to page 8 line37 by the following text:

11.13.19 CS specific service flow encodings

11.13.19.1 CS specification

Type	Length	Value	Scope
[145/146].28	1	~~0: No CS~~ 0: GPCS (Generic Packet Convergence Sublayer) 1: Packet, IPv4 2: Packet, IPv6 3: Packet, 802.3/Ethernet 4: Packet, 802.1Q VLAN 5: Packet, IPv4 over 802.3/Ethernet 6: Packet, IPv6 over 802.3/Ethernet 7: Packet, IPv4 over 802.1Q VLAN 8: Packet, IPv6 over 802.1Q VLAN 9: ATM 10: Packet, 802.3/etherneta with ROHC header compression 11: Packet, 802.3/ethernetb with ECRTP header compression 12: Packet, IP2 with ROHC header compression 13: Packet, IP2 with ECRTP header compression 10 14~255: reserved	DSx-REQ

11.13.19.2 CS Parameter encoding rules

CST	CS
98	~~No CS~~ GPCS (Generic Packet Convergence Sublayer)
99	ATM
100	Packet, IPv4
101	Packet, IPv6
102	Packet, 802.3/Ethernet
103	Packet, 802.1Q VLAN
104	Packet, IPv4 over 802.3/Ethernet
105	Packet, IPv6 over 802.3/Ethernet
106	Packet, IPv4 over 802.1Q VLAN
107	Packet, IPv6 over 802.1Q VLAN
108	Packet, IP with header compression (ROHC)
109	Packet, IP with header compression (ECRTP)
110	Packet, IP over 802.3/Ethernet with header compression (ROHC)
111	Packet, IP over 802.3/Ethernet with header compression (ECRTP)

11.13.19.3.4.3 GPCS PROTOCOL_TYPE Encoding

The encoding of the value field is that defined in an Internet Assigned Numbers Authority (IANA) registry.

<TBD: insert reference to registry here>. The protocol type is defined as one byte.

Type	Length	Value	Scope
[145/146].cst.3.21	1	One byte 802.16 protocol number	DSx-REQ, DSx-REP

For a connection using Generic Packet CS, this TLV shall be used to indicate the protocol carried over the connection when the MULTIPROTOCOL_ENABLE is disabled. For other packet CS types, PROTOCOL_-TYPE is not used.

11.13.19.3.4.19 MULTIPROTOCOL_ENABLE Encoding

This parameter is used to indicate whether or not multiple upper layer protocol data packets are allowed to be transported over the same connection when the Generic Packet Convergence sublayer (GPCS) is used. This parameter only applies to GPCS.

If enabled, the connection can carry multiple upper layer protocols and the PROTOCOL_TYPE field must be prepended to each upper layer data packets for the corresponding CID. See section 5.2.8.2 for the GPCS PDU format. If disabled, a connection must establish the single PROTOCOL_TYPE in use by exchanging the PROTOCOL_TYPE TLV in the DSx messages as specified in section 11.13.19.3.4.3.

Type	Length	Value	Scope
[1445/146].cst.3.20	1	0: MULTIPROTOCOL_-ENABLE is disabled, default value. 1: MULTIPROTOCOL_-ENABLE is enabled, SDU is prepended with PROTOCOL_TYPE value described in section 11.13.19.3.4.3 2-255: Reserved for future use	DSx-REQ, DSx-REP