



Curso Multimedia Home Platform 1.1.2

MHP. Protocolos de Transporte

Broadcast

Interactive

Curso Multimedia Home Platform 1.1.2

Copyright 2008 © Enrique Pérez Gil

Licensed under the ***Creative Commons Attribution-Non-Commercial-No Derivative Works 3.0 Unported License***. You may not use this file except in compliance with the License. You may obtain a copy of the License at:

<http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode>

This is a human-readable summary of the License applied:

(<http://creativecommons.org/licenses/by-nc-nd/3.0/>)

You are free to Share, to copy, distribute and transmit the work **Under the following conditions:**

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial.** You may not use this work for commercial purposes.
- **No Derivative Works.** You may not alter, transform, or build upon this work.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.

Introducción

- Para poder comunicarse con el mundo exterior MHP ha de ser capaz de hacerlo a través de sistemas usando determinados protocolos.
- Cuando el soporte consista en servicios **Broadcast** estos protocolos se apoyarán en **DSMCC-U-U, Data Carousel, Objects Carousels** y en el caso de **Multiprotocol Encapsulation** lo harán para ofrecer protocolos **IP** (unidireccionales generalmente)
- Cuando el soporte consista en servicios **Interactivos** se ofrecerán protocolos que se apoyarán en **IP** (bi-direccionales generalmente)
- En la especificación MHP se distinguen como protocolos **Broadcast e Interaction**.

Documentos a tener a mano

- MHP 1.1.2 A0068r1 specs
- ISO- 13818-6: DSMCC
- ISO- 13818-1: MPEG-2
- EN 301 192: Digital Video Broadcasting (DVB); DVB specification for data broadcasting
- ETSI TR 101 202: Digital Video Broadcasting (DVB); Implementation guidelines for Data Broadcasting
- ETSI TR 101 162: Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems

Broadcast

- Con **Broadcast** se refiere a las posibilidades de comunicación existentes utilizando como soporte la señal emitida, ya sea terrestre, satélite o cable. Podemos dividirlos a su vez en 3 grupos:
 - **Streams MPEG-2**: En este grupo se encuentra la recepción de **Streams MPEG-2**, los cuales como ya hemos visto “un poco” nos permiten recibir datos, video, audio, tablas de información (qué va en las tramas, de qué aplicaciones existen...etc...etc)....
 - **DSMCC**: consiste en una especificación amplísima orientada a permitir la comunicación en entornos heterogéneos, ya sea unidireccional o bidireccional (dependiendo de las posibilidades del contexto).
 - **IP**: protocolos sobre IP apoyándose en el denominado **Multiprotocol Encapsulation (MPE)**

Broadcast

- Esquema de protocolos Broadcast

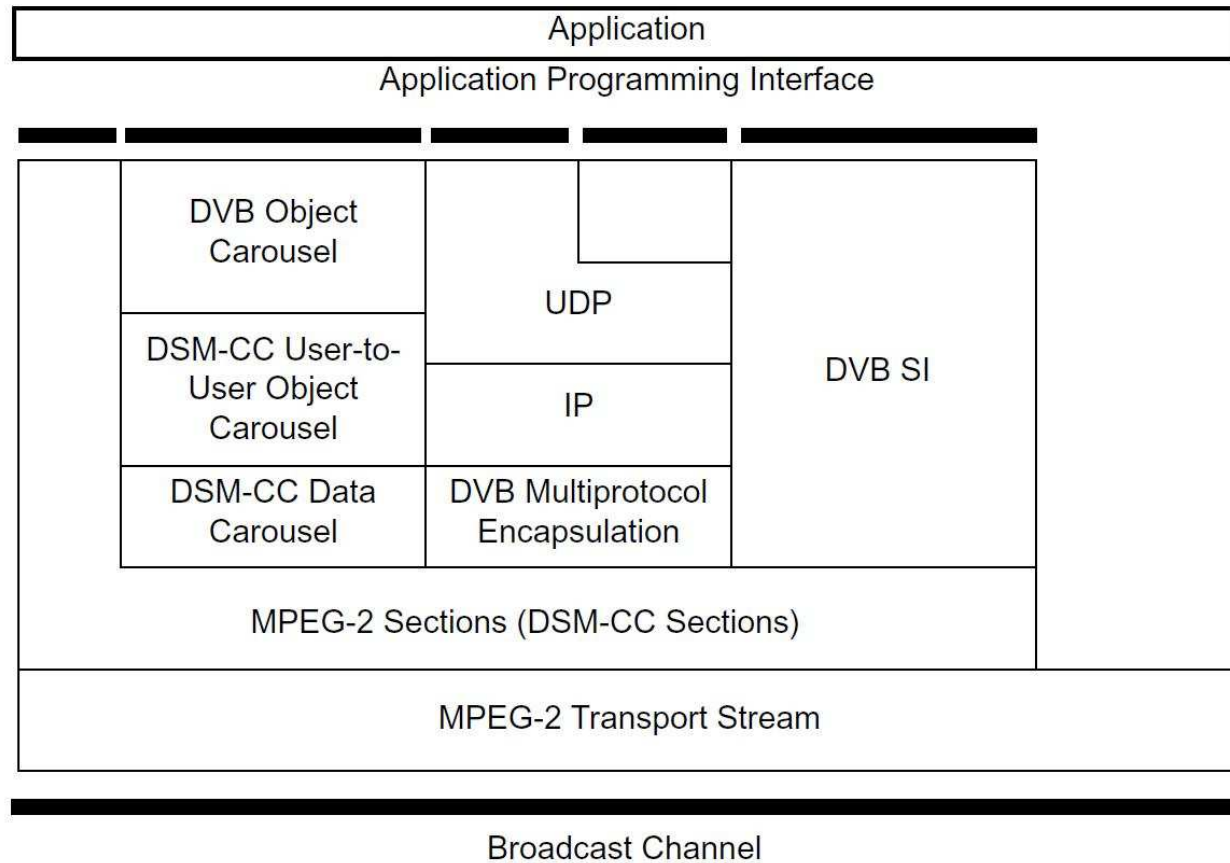


Figure 8: Broadcast Channel Protocol Stack

A0068r1

Broadcast

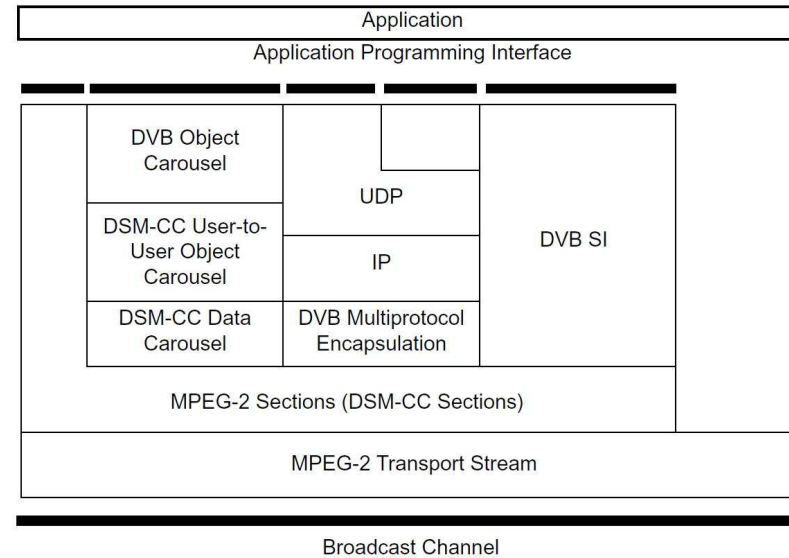


Figure 8: Broadcast Channel Protocol Stack

A0068r1

Resumidamente, a bajo nivel:

- Sobre Broadcast disponemos de transporte sobre PES packets para tramas de video y audio y MPEG-2 Sections para DVB SI.
- Todo lo que es DSMCC va sobre DSMCC Sections, que es una adaptación sobre las MPEG-2 Sections.

Broadcast

- Tal y como se enumeran en las Specs los Protocolos **Broadcast** son los siguientes:
 - **MPEG-2 Transport Stream**
 - **DVB Service Information.** Datos sobre el contenido. Sobre **MPEG-2 Private Sections.**
 - **DSM-CC User-to-User Object Carousel:** Ficheros, Directorios, Events. Sobre **DSMCC Data Carousels.**
 - **DSM-CC Data Carousel:** Paquetes de Datos sin Significado. Sobre **DSMCC Sections:** formato propio apoyándose en la Private Section de MPEG-2.
 - **DVB Multiprotocol Encapsulation (MPE),** basado en **DSMCC Private data** ofrece soporte de IP. Solo se soporta Multicast IP (no unicast)
 - **DSMCC Private Data:** Streams de tipo DSMCC Sections con contenido es definido ad-hoc apoyándose en el formato private data “libre”.
 - **MPEG-2 Private Sections:** Formato ideal para transmisión de Datos sobre ES.
 - **Internet Protocol (IP):** tal y como está definido por DARPA. IETF RFC 768. Sobre **MPE.**
 - **User Datagram Protocol (UDP):** tal y como está definido por DARPA. IETF RFC 791. Sobre **MPE.**
 - **IP signalling: IP Notification Table: tabla INT.** Permite definir destinatarios basados en IP/MAC y acciones asociadas a los destinatarios. **La INT se transmite como un SERVICE!!!**

Interactive

- Se refiere a aquellos protocolos que podemos usar por el canal de retorno, esto es, cuando establecemos comunicación por un contexto distinto al Broadcast.
- Según MHP 1.1.2, A0068r1 en la figura observamos el “Protocol Stack” para Interactive protocols.

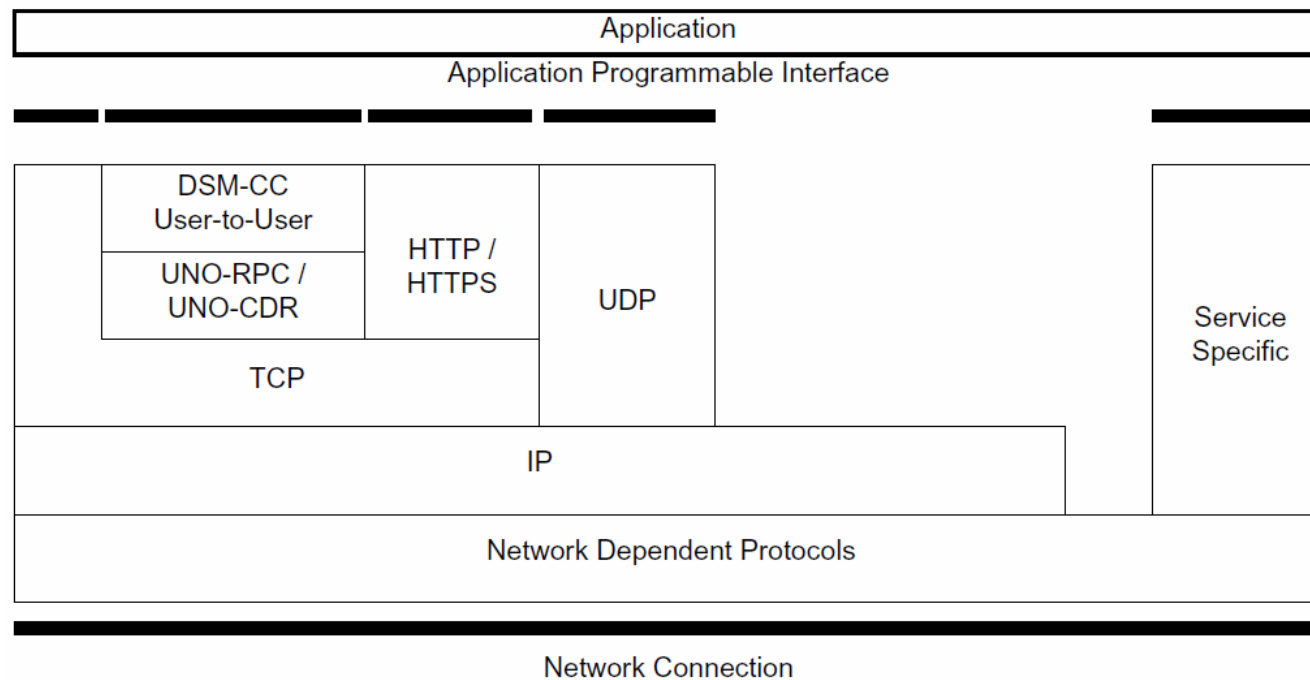


Figure 9: Interaction Channel Protocol Stack

Interactive

- La relación de protocolos según MHP 1.1.2 A068r1 es la siguiente:
 - Protocolos de comunicación de red: CATV, PSTN/ISDN, DECT, GSM, LMDS SMATV
 - Internet Protocol (IP)
 - Transmission Control Protocol (TCP)
 - UNO-RPC: Internet Inter-ORB Protocol (IIOP), CORBA/IIOP.
 - UNO-CDR: Common Data Representation, CORBA/IIOP.
 - DCM-CC User to User
 - Hypertext Transfer Protocol (HTTP)
 - User Datagram Protocol (UDP)
 - DNS

Interactive

- ¿ DSMCC ? ¿ UNO-RPC ? ¿ UNO-CDR ? CORBA ? Que no cunda el pánico.
 - En primer lugar hemos de recordar que DSMCC está definido de manera independiente de la infraestructura de red, de manera que puede usarse tanto en una red de un solo sentido (Broadcast) como en otra de doble (TCP/IP, Internet...).
 - DSMCC se apoya en UNO-RPC (y UNO-CDR) para su definición, estamos hablando de CORBA/IIOP.
 - En cuanto a la aparición de DSMCC-U-U como base de protocolos IP en servicios bi-direccionales, no lo vamos a ver. En cualquier caso, en el supuesto de estar disponible esta bi-dirección, y de usarse se entiende que sería transparente.
- Respecto a los protocolos de comunicación de red, no los vamos a ver pero están ahí, finalmente son la “cacharrería” y los APIS gracias a los cuales podemos comunicarnos a través de tipos de redes variopintas: CATV, PSTN/ISDN, DECT, GSM, LMDS SMATV

Interactive

- Os estaréis preguntando ¿ **qué son**

File System Implemented only Via de Interaction Channel e

Hybrid between Broadcast Stream e Interaction Channel

?

- Estos son dos “protocolos” o especificaciones para bajarnos aplicaciones a través del Return Channel o bien mezcándolo con el Broadcast.
- Sólo se usan para este propósito y no consisten en APIS para desarrolladores. (aunque sí para Deployment). Ambos se cubren en el Tema **App Signalling**.

Interactive

- Además de todo lo anterior, recordemos los perfiles MHP en cuanto al soporte de protocolos de comunicación. ¿ **Sobre qué vamos a trabajar mayormente ?**
- Broadcast:
 - **MPEG-2,**
 - **DSMCC-U-U(ficheros...)**
- Interactive:
 - **TCP/IP**
 - **UDP**
 - **HTTP 1.0, DNS, HTTPS**

Area	Specification	Enhanced Broadcast Profile 2	Interactive Broadcast Profile 2	Internet Access Profile 2
Broadcast channel protocols				
	6.2.2, "MPEG-2 Sections"	M	M	M
	6.2.5, "DSM-CC User-to-User Object Carousel"	M	M	M
	IP Multicast stack based on: 6.2.6, "DVB Multiprotocol Encapsulation", 6.2.7, "Internet Protocol (IP)" 6.2.8, "User Datagram Protocol (UDP)" 6.2.10, "IP signalling"	O	Ro	M
Interaction channel protocols				
TCP/IP	6.3.3, "Transmission Control Protocol (TCP)" 6.3.2, "Internet Protocol (IP)"	-	M	M
UDP/IP	6.3.2, "Internet Protocol (IP)" 6.3.9, "User Datagram Protocol (UDP)"	-	M	M
DSM-CC U-U RPC	6.3.4, "UNO-RPC" 6.3.5, "UNO-CDR" 6.3.6, "DCM-CC User to User"	-	O	O
HTTP	6.3.7.1, "HTTP 1.1"	-	O	O
HTTP	6.3.7.2, "MHP profile of HTTP 1.0"	-	M	M
DNS	6.3.10, "DNS"	-	M	M
HTTPS	6.3.7.3, "HTTPS"	-	M	M
Interaction Channel File System	6.4.1, "File system implemented only via the interaction channel"	-	M	M
DSMCC / HTTP hybrid	6.4.2, "Hybrid between broadcast stream and interaction channel"	-	M	M

Qué vamos a ver en este capítulo

- De la parte de **Broadcast** en cuanto a **MPEG-2** se refiere nos interesa todo lo que sea poder acceder a la información transmitida en los Streams, lo que llamamos **System Information**, para lo cual usaremos 3 mecanismos: **DVB SI**, **JavaTV SI** y **Section Filtering**; permitiéndonos este último analizar Streams de datos a bajo nivel. Estos 3 APIs se ven en detalle por separado en tres capítulos.
- También bajo el paraguas de MPEG-2 la gestión de los Services (con sus tramas Audio/Video, aplicaciones,...) se verá en detalle en capítulos por separado: Service Context, Java Media Framework, Tuning, Interaction Channels...
- Al margen de MPEG-2 disponemos de **DSMCC**, respecto al cual veremos detalladamente más adelante en sendos capítulos lo relacionado con la parte de **DSMCC-U-U**: el envío de **Estructuras de Directorios y Ficheros** al receptor y lo relacionado con **Stream Events** y **NPT** (Normal Play Time)
- En este capítulo hablaremos con más detalle de los protocolos relacionados con IP , tanto de la parte de Broadcast como de Interactive.

Broadcast. Protocolos IP

- IP signalling: Permite definir destinatarios basados en IP/MAC y acciones asociadas a los destinatarios. Sobre **MPE**.
- User Datagram Protocol (UDP): tal y como está definido por DARPA. IETF RFC 791. Sobre **IP/MPE**
- Internet Protocol (IP): tal y como está definido por DARPA. IETF RFC 768. Sobre **MPE**
- DVB Multiprotocol Encapsulation, basado en DSMCC Private data protocol ofrece soporte de IP. Solo se soporta Multicast IP (no unicast)

IP Signalling

- A partir de la versión MHP 1.0.3 y MHP 1.1.1 la forma en que se pueden describir los destinatarios es mediante lo que se llama **INT**: IP/MAC Notification Table.

Brevemente. Modos de Transmisión a bajo nivel

- Antes de entrar en harina con cada protocolo recordemos los mecanismos a bajo nivel que se usarán.
- Los Streams MPEG permiten transportar, además de Audio y Video, Sistemas de Ficheros, información de CA, datos para permitir la descryptación, subtítulos, referencias de tiempo...PSI...
- La información de un **TS** viaja multiplexada en su más bajo nivel en paquetes de **188 bytes / packet**. Los Streams de Video y Audio utilizan por encima un esquema de transmisión ideal para datos que requieren sincronismo denominado **PES: Packetized Elementary Stream**, y de igual manera la información de **PSI** utiliza lo que se llama **Private Sections**.
- Si leemos el apartado 4.2 del documento ETSI TR 101 202 V1.2.1, **Implementation guidelines for Data Broadcasting**, leemos lo siguiente:

Brevemente. Modos de Transmisión a bajo nivel

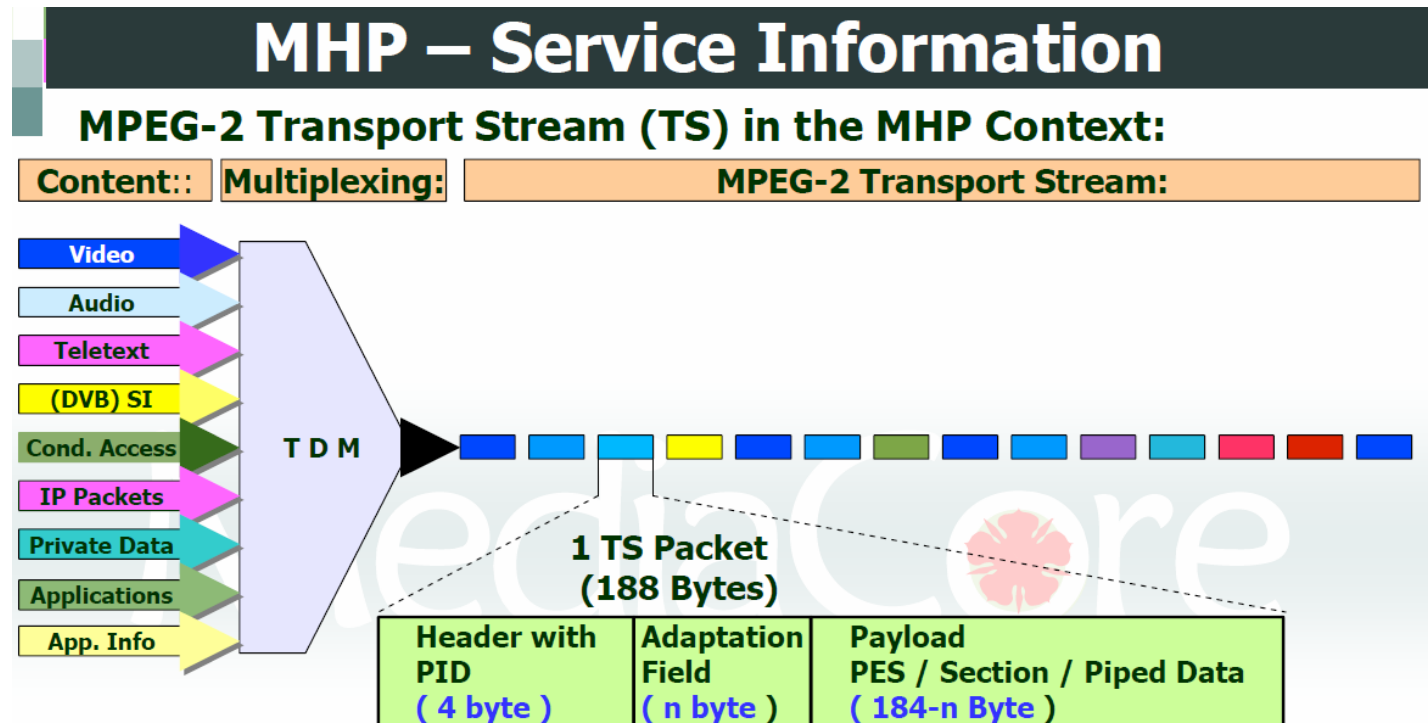
4.2 , ETSI TR 101 202 V1.2.1

Generally data of any kind of protocols are transmitted in packetized form ("datagrams"). These datagrams may have different length. If the data are not packetized or the packetization method is irrelevant or hidden to the DVB transmission chain the most appropriate way of transmission is the **Data Pipe (see EN 301 192 [1], clause 4)**.

On the layer of MPEG-2 Transport Stream data are transmitted within packets with a fixed length of **188 bytes (184 bytes payload)**, therefore datagrams of higher layers must be fragmented at the transmission side and be re-assembled at the reception. For fragmentation of the datagrams there are three possible ways (see also figure 4.1):

- **Private mechanisms based on the Data Pipe.**
- **MPEG-2 Packetized Elementary Streams (PES).**
- **MPEG-2 Sections.**

Brevemente. Modos de Transmisión a bajo nivel



- **MPEG-2 packets can contain :**

Video, Audio, Teletext, Data streaming (13818-1)

DSM-CC Sections (data carousel, object carousel, SI-tables, etc) (13818-6)

DVB Data Piping

Brevemente. Modos de Transmisión a bajo nivel

4.2 , ETSI TR 101 202 V1.2.1

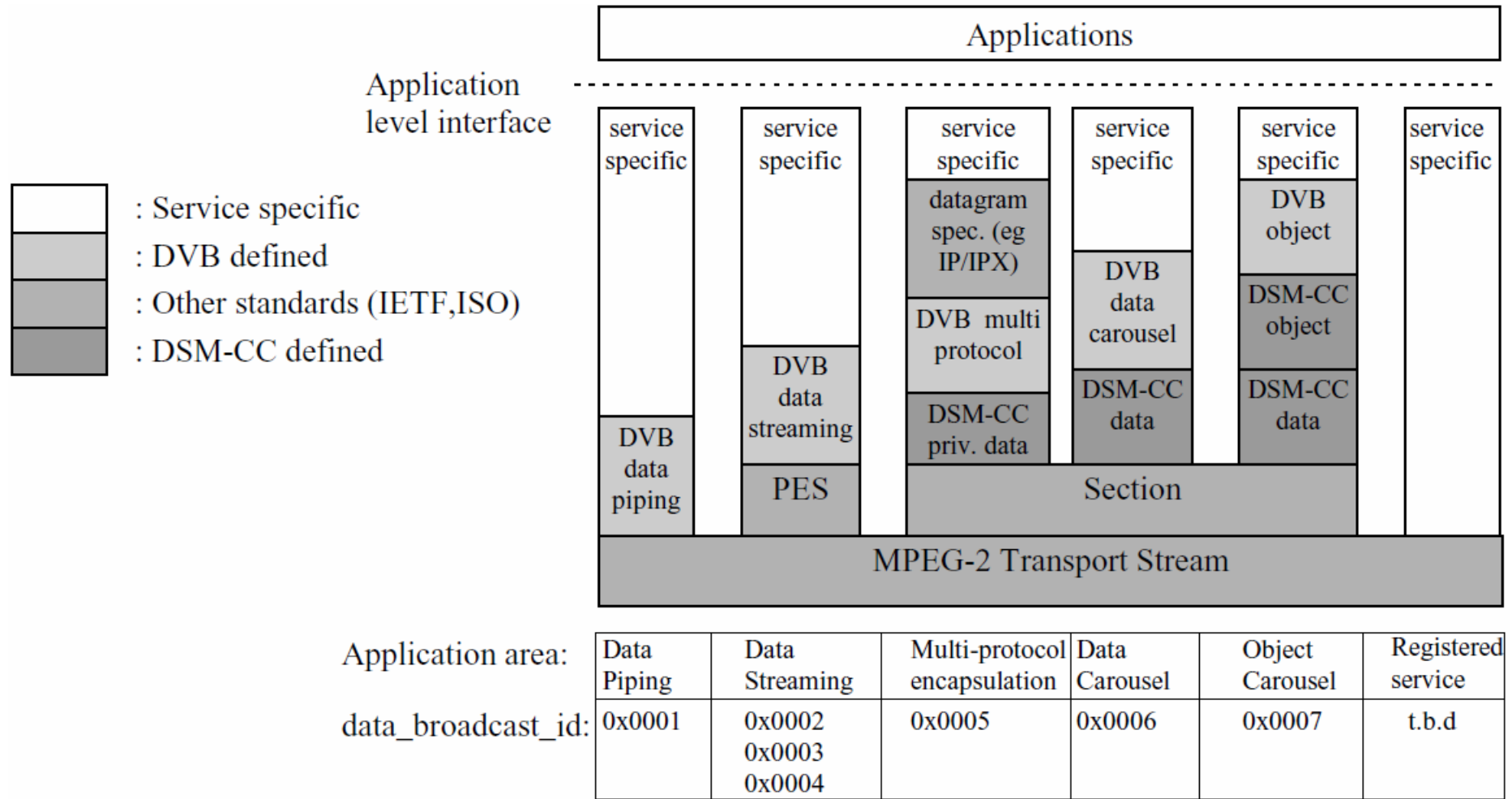
MPEG-2 PES provides a mechanism to transmit datagrams of variable size with a maximum length of **64 kbytes**. Additionally it provides the facility to **synchronize different data streams accurately** (as used in MPEG for synchronization of Video and Audio), therefore it was chosen by DVB for the transmission of **synchronous and synchronized but also asynchronous data streams** (see EN 301 192 [1], clauses 5 and 6).

MPEG-2 Sections can be used to transmit datagrams of variable size with a maximum length of **4 kbytes**. The transmission is **asynchronous**. **MPEG-2 Sections are built in a way that MPEG-2 demultiplexers available on the market** can filter out single sections in hardware which may reduce the required software processing power of the receiver. This is the main reason why the **MPEG-2 Sections have been chosen as the mechanism for the transmission of encapsulated protocols and data carousels**.

Brevemente. Modos de Transmisión a bajo nivel

- Como veis, por debajo tenemos el formato es MPEG-Sections para la transmisión de gran parte de los protocolos, incluyendo DVB SI.
- Para la transmisión de Video/Audio se reserva PES.
- Encima, como vemos, se describen numerosos protocolos que se apoyan en estos.

Protocolos y Formatos (TR 101 202)



IMPORTANTE: data_broadcast_id

- El **data_broadcast_descriptor** tiene el formato apuntado (EN 300 468) siendo su tag = 0x64, y sirve para ofrecer información acerca del tipo de Protocolo empleado que viaja en un ES.
- El **data_broadcast_descriptor** puede aparecer en la SDT y EIT mientras que una versión reducida lo hace en la **PMT: data_broadcast_id_descriptor (tag = 0x66)**. En el primer caso “apunta” a un stream (component_tag) y en el segundo lo califica (loop).
- El campo **data_broadcast_id** nos dice el modelo seguido para transmitir datos en el ES indicado. Por ejemplo, valdrá **0x0005** para indicar que este descriptor de data nos lleva a un Stream de tipo **Multiprotocol Encapsulation**. Ved tipos existentes en la siguiente slide.
- Interesante el tema del data_broadcast_descriptor y el data_broadcast_id. Ved EN 300 192, EN 300468 y TR 101 202.

Syntax	Number of bits	Identifier
data_broadcast_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
data_broadcast_id	16	uimsbf
component_tag	8	uimsbf
selector_length	8	uimsbf
for (i=0; i<selector_length; i++){		
selector_byte	8	uimsbf
}		
ISO_639_language_code	24	bslbf
text_length	8	uimsbf
for (i=0; i<text_length; i++){		
text_char	8	uimsbf
}		
}		

Syntax	Number of bits	Identifier	
data_broadcast_id_descriptor(){			
descriptor_tag	8	uimsbf	
descriptor_length	8	uimsbf	
data_broadcast_id	16	uimsbf	TR 101 162
for(i=0; i < N;i++){			
id_selector_byte	8	uimsbf	
}			
}			

Valores de **data_broadcast_id** reservados por DVB y para protocolos propietarios.

Data broadcast specification	Data_broadcast_id
Reserved for future use	0x0000
Data pipe	0x0001
Asynchronous data stream	0x0002
Synchronous data stream	0x0003
Synchronised data stream	0x0004
Multi protocol encapsulation	0x0005
Data Carousel	0x0006
Object Carousel	0x0007
DVB ATM streams	0x0008
Higher Protocols based on asynchronous data streams	0x0009
Reserved for future use by DVB	0x000A-0x00FF
Reserved for registration	0x0100-0xFFFFE
Reserved for future use	0xFFFF

Data_broadcast_id	Data broadcast specification name
0x0100	Eutelsat Data Piping
0x0101	Eutelsat Data Streaming
0x0102	SAGEM IP encapsulation in MPEG-2 PES packets
0x0103	BARCO Data Broadcasting
0x0104	CyberCity Multiprotocol Encapsulation (New Media Communications Ltd.
0x0105	CyberSat Multiprotocol Encapsulation (New Media Communications Ltd.
0x0106	The Digital Network
0x0107	OpenTV Data Carousel
0x0108	Panasonic
0x0109	MSG MediaServices GmbH
0x0110	Televizija Polsat
0x0111	UK DTG
0x0112	SkyMedia
0xB BBBB	Bertelsmann Broadband Group
0xB BBB1	BBG Data Caroussel
0xB BBB2	BBG Object Caroussel

Broadcast. IP Signalling

- La **INT** es una tabla que se puede transmitir de la misma forma que las **Tablas de Información de DVB SI**, y ser manejada **en el SI como si fuera un Service**. EN 301 192:
 - The IP/MAC Notification Table (INT) is signalled as a **DVB service (SDT Table)**. The **PMT** of the transport stream (debería ser Service) carrying the *INT* shall contain the **data_broadcast_id_descriptor(tag=0x66)** with the data_broadcast_id of **0x000B** to indicate the elementary stream used for the *IP/MAC Notification* Table.
- **En la tabla obligatoria SDT donde se describen los Services del TS, para cada Service se dice su tipo, y uno de los tipos es 0x0C: Data Broadcast Service.** Vemos en la siguiente Slide el descriptor de tipo.
- También se puede informar a nivel de **NIT** o **Bouquet** (Ved EN301 192) de donde se encuentra un **TS/Service de tipo INT** mediante linkage descriptors (EN 301192). El objeto de disponer de acceso a una **INT a nivel de Network o Bouquet**, puede ser por ejemplo si deseamos transmitir datos a todos los suscriptores de TVE.

SDT

- Mediante el **service_descriptor** nos indica que **Tipo de Servicio** es cada uno.
- Notad:
 - 0x01: Digital Tv
 - 0x02: Digital Radio
 - 0x10: DVB MHP Service
 - 0x0C: Data Broadcast Service**

Syntax	Number of bits	Identifier
service_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
service_type	8	uimsbf
service_provider_name_length	8	uimsbf
for (i=0;i<N;I++){		
char	8	uimsbf
}		
service_name_length	8	uimsbf
for (i=0;i<N;I++){		
Char	8	uimsbf
}		
}		

service_type	Description
0x00	reserved for future use
0x01	digital television service (see note 1)
0x02	digital radio sound service (see note 2)
0x03	Teletext service
0x04	NVOD reference service (see note 1)
0x05	NVOD time-shifted service (see note 1)
0x06	mosaic service
0x07	reserved for future use
0x08	reserved for future use
0x09	reserved for future use
0x0A	advanced codec digital radio sound service
0x0B	advanced codec mosaic service
0x0C	data broadcast service
0x0D	reserved for Common Interface Usage (CENELEC EN 50221 [38])
0x0E	RCS Map (see EN 301 790 [7])
0x0F	RCS FLS (see EN 301 790 [7])
0x10	DVB MHP service
0x11	MPEG-2 HD digital television service
0x12 to 0x15	reserved for future use
0x16	advanced codec SD digital television service
0x17	advanced codec SD NVOD time-shifted service
0x18	advanced codec SD NVOD reference service
0x19	advanced codec HD digital television service
0x1A	advanced codec HD NVOD time-shifted service
0x1B	advanced codec HD NVOD reference service
0x1C to 0x7F	reserved for future use
0x80 to 0xFE	user defined
0xFF	reserved for future use

NOTE 1: MPEG-2 SD material should use this type.

NOTE 2: MPEG-1 Layer 2 audio material should use this type.

SDT

- Un apunte respecto a Canales de Aplicaciones: **0x10: DVB MHP Service**. Strong 5510 MHP 1.1.2



Broadcast. IP Signalling

- Formato de la INT

Table 13: Syntax of the IP/MAC_notification_section

Name	Number of bits	Identifier	Remarks
IP/MAC_notification_section() {			
table_id	8	uimsbf	0x4C
section_syntax_indicator	1	bslbf	1b
reserved_for_future_use	1	bslbf	1b
reserved	2	bslbf	11b
section_length	12	uimsbf	
action_type	8	uimsbf	see table 14
platform_id_hash	8	uimsbf	
reserved	2	bslbf	11b
version_number	5	uimsbf	
current_next_indicator	1	bslbf	1b
section_number	8	uimsbf	
last_section_number	8	uimsbf	
platform_id	24	uimsbf	
processing_order	8	uimsbf	0x00
platform_descriptor_loop()			
for (i=0, i<N1, i++) {			
target_descriptor_loop()			
operational_descriptor_loop()			
}			
CRC_32	32	rpchof	
}			

EN 301 192

Broadcast. IP Signalling

- Mediante la **INT** es posible definir rangos de destinatarios (multicast) en función de IP, MAC, smartcard...e incluso acciones a realizar sobre ellos. Fijaos en las posibilidades de descriptores para cada loop de la tabla anterior (EN 301 192)

Table 19: INT descriptors

Descriptor	Tag Value	Allowed in Loop		
		Platform	Target	Operational
reserved	0x00			
target_smartcard_descriptor	0x06		*	
target_MAC_address_descriptor	0x07		*	
target_serial_number_descriptor	0x08		*	
target_IP_address_descriptor	0x09		*	
target_IPv6_address_descriptor	0x0A		*	
IP/MAC_platform_name_descriptor	0x0C	*		
IP/MAC_platform_provider_name_descriptor	0x0D	*		
target_MAC_address_range_descriptor	0x0E		*	
target_IP_slash_descriptor	0x0F		*	
target_IP_source_slash_descriptor	0x10		*	
target_IPv6_slash_descriptor	0x11		*	
target_IPv6_source_slash_descriptor	0x12		*	
IP/MAC_stream_location_descriptor	0x13	*		*
ISP_access_mode_descriptor	0x14	*		*
telephone_descriptor	0x57	*		*
private_data_specifier_descriptor	0x5F	*	*	*
user private	0x80 to 0xFE			
reserved	0xFF			

Broadcast. IP Signalling

- ¿ Y donde están los datos que debe recibir el destinatario ? Si nos fijamos el descriptor de la INT, el **operational_descriptor_loop()** será el encargado de indicarnos donde está el DSMCC Stream (**tipo 0x0D, private data**) que contiene la información que se debe “enchufar” a, por ejemplo, el rango de la IP/port indicado.
- Tipos de DSMCC Streams:
 - 0x00-0x09** ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined
 - 0x0A** Multi-protocol Encapsulation
 - 0x0B** DSM-CC U-N Messages, Objects or Data Carousels
 - 0x0C** DSM-CC Stream Descriptors
 - 0x0D DSM-CC Sections (any type, including private data)**
 - 0x0E - 0x7F** ITU-T Rec. H.222.0 | ISO/IEC 13818-1 reserved
 - 0x80 - 0xFF** User private
- El descriptor que nos dice donde se encuentra el Stream es :
IP/MAC stream_location_descriptor
- Veamos a continuación como se almacena la información de los datagram_packets en el DSMCC Stream.

Broadcast. IP Signalling

- Formato de los paquetes que se mandan en el Stream DSMCC 0x0D
 - Table_id = **0x3E** (DSM-CC sections with private data (ISO/IEC 13818-6 [5])).
- Como veremos a continuación, este DSMCC Stream es el usado también en el **DVB Multiprotocol Encapsulation, ¿ entonces ?... >>>>** Lo que se ha hecho con INT es una forma mucho más eficiente de “enrutar” los paquetes; además esta definición de IP Signalling fue acordada posteriormente a la de DVB Multiprotocol Encapsulation.

Syntax	Number of bits	Mnemonic
datagram_section() {		
table_id	8	uimbsf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
section_length	12	uimbsf
MAC_address_6	8	uimbsf
MAC_address_5	8	uimbsf
reserved	2	bslbf
payload_scrambling_control	2	bslbf
address_scrambling_control	2	bslbf
LLC_SNAP_flag	1	bslbf
current_next_indicator	1	bslbf
section_number	8	uimbsf
last_section_number	8	uimbsf
MAC_address_4	8	uimbsf
MAC_address_3	8	uimbsf
MAC_address_2	8	uimbsf
MAC_address_1	8	uimbsf
if (LLC_SNAP_flag == "1") {		
LLC_SNAP()		
} else {		
for (j=0;j<N1;j++) {		
IP_datagram_data_byte	8	bslbf
}		
}		
if (section_number == last_section_number) {		
for (j=0;j<N2;j++) {		
stuffing_byte	8	bslbf
}		
}		
if (section_syntax_indicator == "0") {		
checksum	32	uimbsf
} else {		
CRC_32	32	rpchof
}		
}		

Broadcast. IP Signalling

- Por tanto existen dos formas de enrutar paquetes pero esta es la recomendada. Veamos la otra.

Broadcast. DVB Multiprotocol Encapsulation. Datagramas

- Usado para enviar **Datagramas** a destinatarios. Los datagramas también se almacenan en Streams cuyo tipo es **0x0D**, el tipo “multi usos” de DSMCC.
- En este caso el mecanismo que el **Data Broadcast Service** usa para enrutar los paquetes es distinta a la **INT**: lo hará incluyendo uno o más **data broadcast descriptors** en la **SDT** con la siguiente información:
 - Tag apuntando a un **DSMCC Stream de tipo 0x0D** mediante un **component_tag** (ya lo vimos en APP Signalling: el valor del **component_tag** es el mismo que el del campo **component_tag** del **stream_identifier_descriptor** que lleva asociado el Stream en su definición en la PMT).
 - Información de routing de la forma siguiente.

Broadcast. DVB Multiprotocol Encapsulation (EN 301 192, EN 300 468)

- El campo selector_byte del **data_broadcast_descriptor** contendrá la información de enrutado descrita: multiprotocol_encapsulation_info. Consiste básicamente en filtrado de MACs/IPs

Syntax	Number of bits	Identifier
data_broadcast_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
data_broadcast_id	16	uimsbf
component_tag	8	uimsbf
selector_length	8	uimsbf
for (i=0; i<selector_length; i++){		
selector_byte	8	uimsbf
}		
ISO_639_language_code	24	bslbf
text_length	8	uimsbf
for (i=0; i<text_length; i++){		
text_char	8	uimsbf
}		
}		
}		

Syntax	Number of bits	Mnemonic
multiprotocol_encapsulation_info () {		
MAC_address_range	3	uimsbf
MAC_IP_mapping_flag	1	bslbf
alignment_indicator	1	bslbf
reserved	3	bslbf
max_sections_per_datagram	8	uimsbf
}		

Broadcast

En resumen:

- **IP/MAC Signalling** en la SDT (o NIT o BOUQUET) se nos indica que hay un Service de tipo Data Broadcast dentro del cual hay un Stream que contiene una tabla, la INT, con la información del enrutamiento. Además nos indican en qué ES de tipo DSMCC Private Data se encuentra la información.
- **DVB Multiprotocol Encapsulation**, en la SDT se nos indica que hay un Service de tipo Data Broadcast, y en la misma SDT para dicho Service, se incluyen N descriptores del tipo **data_broadcast_descriptor**, cada uno de los cuales nos indica a quienes han de entregarse los datos y en qué ES del tipo DSMCC Private Data residen estos

Interactive

- HTTP 1.1 es opcional. Podemos saber si es soportado leyendo la propiedad “mhp.option.http.1.1”

“MHP Profile of HTTP 1.0”

- Se soporta HTTP 1.0 tal y como se define en la RFC 1945.
- HTTP 1.0 tal y como se especifica en RFC 1945 no soporta el token “keep-alive” que es el indicador para usar **Persistent Connections**.
- En MHP es obligatorio soportar **Persistent Connections** en los términos que veremos a continuación.
- Por otro lado MHP dice que siempre que una implementación trabaje con un profile que deba soportar HTTP 1.0, esta es libre de soportar versiones superiores de HTTP, como HTTP 1.1, siempre que mantenga compatibilidad hacia atrás con el soporte de **Persistent Connections** detallado.

Persistent Connections. Un poco de teoría.

<http://java.sun.com/j2se/1.5.0/docs/guide/net/http-keepalive.html>

- Consiste en usar la misma conexión TCP para efectuar muchas llamadas HTTP.
- Las ventajas son obvias:
 - Ahorro de tiempo en establecimientos de conexión TCP.
 - Ahorro de puertos abiertos en cliente que pueden provocar errores en aplicaciones que hagan uso intensivo de comunicaciones.
 - Cuando trabajamos con SSL/TLS el ahorro de tiempo es muy importante al reducir las negociaciones para establecer la conexión segura.
- En HTTP 1.1 el comportamiento por defecto del servidor es el de mantener las conexiones, incluso después de “Error Responses”, de forma que los clientes han de actuar en consecuencia si quieren aprovechar esta circunstancia. Por eso al leer una petición HTTP y recibir un “Error Response” conviene leer el errorStream.

Persistent Connections. Un poco de teoría. (y 2)

- ¿ Qué hace que una conexión sea reutilizable ? Las conexiones TCP son por naturaleza de tipo stream, por lo que necesitamos un mecanismo que nos permita saber con exactitud el final de una petición y el comienzo de otra, es decir, que para **TODOS LOS MENSAJES** en la conexión, necesitamos saber el tamaño del mensaje: **Content-Length**.

MHP Persistent Connections

- En RFC 2068 se explica de manera informal la manera en que se deben implementar HTTP 1.0 **persistent connections**.
- Cuando el Terminal efectúe la conexión al servidor enviará el token:

Connection: Keep-Alive

Entonces el servidor responderá con un token Keep-Alive y se establecerá la conexión persistente.

MHP Persistent Connections

Keep-Alive-header

- Una vez que el token **Connection : Keep-Alive** ha sido enviado tanto en una request como en una response, el **header** Keep-Alive también puede incluirse. El Header Keep-Alive toma la forma:

Keep-Alive-header = "Keep-Alive" ":" 0# keepalive-param

keepalive-param = param-name "=" value

- Donde la notación "0#" indica que el campo "keepalive-param" se puede repetir 0 o n veces separado por comas, si hay mas de uno.
- De todos modos, el header Keep-Alive **es opcional**, y en el contexto MHP no se define ningún parámetro, de manera que con respecto a MHP no hay nada relevante que decir. Únicamente explicar que si el header Keep-Alive se envía entonces el header Connection debe estar, si no, se ignorará.

MHP Persistent Connections

Proxies

- En la especificación previene del envío del **Connection:keep-alive** token a HTTP 1.0 proxy servers que no entienden Connection, pues podrían provocar un cuelgue en la conexión.
- Sin embargo por otro lado se lee:
 - Since persistent connections applies to only one transport link, it is important that proxy servers correctly signal persistent/or-non-persistent connections separately with its clients and the origin servers (or to other proxy servers). From a HTTP client or server's perspective, as far as persistence connection is concerned, **the presence or absence of proxy servers is transparent.**

MHP Persistent Connections

Compatibilidad de Versiones

- Un Servidor HTTP 1.1 puede usar conexiones persistentes con un terminal MHP.
- Pero una conexión persistente con un terminal MHP no puede hacer uso de “chunked transfer coding”, y por lo tanto **es obligatorio usar content-length para marcar los límites de cada mensaje**
- Se recomienda que cuando se usen servidores HTTP 1.1 estos soporten el uso de persistencia HTTP 1.0 cuando los clientes inicien la conexión con HTTP 1.0 incluyendo el **Connection=Keep-Alive** Token.

MHP Persistent Connections

- En la URL <http://java.sun.com/j2se/1.5.0/docs/guide/net/http-keepalive.html> se ofrecen recomendaciones desde el punto de vista de programación para poder usar las persistent Connections.

MHP HTTPS. TLS. Seguridad en el canal de retorno

- Se soportará tal y como se especifica en: **IETF RFC 2818, 2000, "HTTP over TLS", TLS = Transport Layer Security**
- TLS se soporta para la seguridad en el canal de retorno tal y como se especifica en **IETF RFC 2246, 1999, "The TLS Protocol, Version 1.0"**
- La plataforma MHP **NO está obligada a:**
 - Soportar SSL 3.0
 - Poder funcionar como servidor TLS

MHP HTTPS. TLS. Seguridad en el canal de retorno

- La plataforma MHP habrá de:
 - Soportar autenticación de cliente para TLS siempre que la fuente de las claves sea proporcionada a **javax.net.ssl.SSLContext** a través de **javax.net.ssl.KeyManager**
 - Implementar las suites de cifrado siguientes:
 - RSA
 - MD5.
 - SHA-1
 - DES.
 - AES

MHP HTTPS. TLS. Seguridad en el canal de retorno

“Downloading” de Certificados para TLS

- Cuando el terminal establece la conexión con el servidor habrá de comprobar que **al menos un certificado** de la “chain of certificates” que el servidor ha lanzado al cliente es un **trusted certificate**. En un entorno de PC esto equivale a comprobar los bajados con el certificado residente en el PC.
- En un Terminal MHP **una aplicación que se baje por el canal de retorno** puede establecer una sesión TLS. En este escenario la aplicación sabe a qué servidor debe conectarse y también conoce un certificado contra el cual puede comprobar que una determinada “chain of certificates” contiene el certificado en el que confía (trusted certificate).

MHP HTTPS. TLS. Seguridad en el canal de retorno

“Downloading” de Certificados para TLS

- El API utilizado para establecer la sesión es el que se apunta a continuación; el proceso de autenticación del servidor implica la autenticación del “certificate chain” enviado por el TLS Server.
- El API es el “J2ME Security (**JSSE**) Optional Package Specification v1.0 which is found in FP 1.1”. **OJO:** JSSE es opcional en PBP 1.1.
- En aquellos terminales en los que la propiedad **mhp.smartcard.reader** sea “SUPPORTED”, las aplicaciones MHP podrán usar un Provider instalado por una aplicación MHP de la forma que se indica en en anexo AJ de las especificaciones para KeyManagerFactory y para TrustManagerFactory

MHP HTTPS. TLS. Seguridad en el canal de retorno

Uso los Certificados para TLS

- Uno o varios root Certificates pueden ser Broadcasted con la aplicación. Si no es posible autenticar ninguno de los del “chain of certificates” enviado por el TLS Server se lanzará una IO Exception.
- Los certificados se denominarán: `dvb.tls.organisation_id.application_id.x`
 - Donde `organisation_id` y `application_id` son los valores en hexadecimal en minúsculas sin ceros delante, y `x` es un string opcional para diferenciarlos si hay mas de uno
- Los certificados se localizarán en el directorio BASE de la app DVB-J. Donde cuelga la misma.

MHP HTTPS. TLS. Seguridad en el canal de retorno

Uso los Certificados para TLS

- Para que los certificados Broadcasted con la app sean válidos han de someterse al mismo proceso de certificación que el resto de ficheros en ese directorio de la aplicación.
- Cada fichero con un certificado TLS sólo contendrá un certificado, siendo su formato el mismo que se usa para la autenticación de aplicaciones.
- **¿ Y si no se proporcionan Certificados con la aplicación ? Entonces se permitirá a la aplicación conectarse a cualquier servidor, en cuyo caso la aplicación podrá usar el API JSSE para obtener el Certificate Chain y validar lo que considere oportuno. En este caso tanto el nombre como las claves públicas habrán de ser comprobadas por la aplicación para validar la identidad del servidor.**

- **Antes de seguir un poco de teoría.** ¿ qué es lo que va a ocurrir entre el cliente y el servidor...? Aproximadamente lo siguiente:
 - El cliente solicita una página segura (p.e. <https://www.terra.es>).
 - El servidor responde con su public key en su certificado (atestigua que es “él” y que esta es su clave pública).
 - El cliente comprueba que el certificado ha sido emitido por una “Trusted Root Certificate Authority”, que es válido aún y que está relacionado con el site visitado.
 - El cliente utiliza entonces la clave pública recibida para encriptar una clave simétrica aleatoria generada en el momento, y el resto de la información que ha de enviar que a su vez previamente ha sido encriptada usando esta clave simétrica. Todo ello lo envía al servidor.
 - El servidor desencripta todo el mensaje con su clave privada, y usa la clave simétrica para desencriptar el resto de datos.
 - El servidor responde al cliente con la información solicitada encriptada con la clave simétrica.
 - El cliente desencripta la información con su clave simétrica y la usa como crea conveniente...
- **Una clave simétrica** es aquella que sirve para encriptar y desencriptar. Es mucho más rápida de usar que una asimétrica (public/private)

Ejercicios Bloque PROT-1

¿ Por qué no encuentra el protocolo HTTPS ?

¿ Por qué no encuentra el protocolo HTTPS ?

- **JSSE.** en Foundation Profile/jsse1.0.3_04/doc/guide/API_users_guide.html#RefAPI vemos que:
 - com.sun.net.ssl: **The JSSE 1.0.3 reference implementation includes a non-standard API in the...**
NOTA: Por eso si la incluís en el código fuente os compila
 - **Los paquetes:** javax.net, javax.net.ssl, javax.security.cert **son el estándar API**
- **sun.net.x:** son clases de PBP 1.1 de la plataforma y son las que disponen de la implementación de SUN para acceder a determinados protocolos. **OJO:** no hagáis nunca referencia a partes de sun.x: *<http://java.sun.com/products/jdk/faq/faq-sun-packages.html>*
- El problema cuando queremos usar HTTPS es la forma en que lo entiende java.net.URL: por defecto intentará obtener el handler del [protocolo] asumiendo la existencia de la clase siguiente:

sun.net.www.protocol.[protocol].Handler, en nuestro caso sería: sun.net.www.protocol.https.Handler
sun.net.www.protocol suele ser el valor por defecto de la variable de entorno: **java.protocol.handler.pkgs**
- Puesto que no disponemos de ese handler **NO entiende el protocolo.** Podemos intentar dos cosas: La primera establecer la variable **java.protocol.handler.pkgs** y la segunda establecer **java.net.URLStreamHandlerFactory**.

- Vamos con la segunda forma: De lo que se trata es de **establecer NUESTRO *URLStreamHandlerFactory***, de manera que este sepa qué devolver cuando se le pida un Handler para el protocolo https.
- Ahora bien ¿ qué HANDLER ? Es muy frecuente al trabajar con JSSE establecer el de la implementación de referencia de JSSE (ver abajo). Ponemos null para el resto de protocolos para que java.net.URL los tome del lugar por defecto (sun.net.www.protocol....):

```
URL.setURLStreamHandlerFactory(new URLStreamHandlerFactory() {  
    public URLStreamHandler createURLStreamHandler(String protocol) {  
        if (protocol.equals("https"))  
            return new com.sun.net.ssl.internal.www.protocol.https.Handler();  
        else  
            return null;  
    }  
});
```

Ejercicios Bloque PROT-2

- **Problema:** `com.sun.net.ssl.internal.www.protocol.https.Handler` **NO SE encuentra en el runtime del deco.**
- El problema que tenemos en el contexto MHP es que si bien **SI** que está en el deco la librería JSSE pura (`javax.net`, `javax.security`), esta **NO** incluye la implementación de referencia **com.sun...** **a pesar de SI incluirse en el FP1.1-jsse-cdc.** O lo que es peor: a pesar de que en las especificaciones del profile de MHP 1.1.2 para interactive es mandatory el soporte de HTTPS, este no funciona en este deco: Strong 5510 MHP 1.1.2.
- Por lo tanto o bien subimos con la aplicación el paquete **FP1.1-jsse-cdc...** o bien lo bajamos remotamente.
- Vamos a proceder a bajar la clase **com.sun.net.ssl.internal.www.protocol.https.Handler** desde un servidor http en el que publicamos **FP1.1-jsse-cdc:** `refs/FP1.1-jsse-cdc.jar`

Ejercicios Bloque PROT-3

- **Problema:** Al proceder a ejecutar, obtenemos:

java.lang.RuntimeException: **Export restriction: this JSSE implementation is non-pluggable.**

[2#6:2] at sun.net.www.protocol.http.HttpURLConnection.getInputStream(Unknown Source)

[2#6:2] at sun.net.www.protocol.http.HttpURLConnection.getHeaderField(Unknown Source)

[2#6:2] at java.net.HttpURLConnection.getResponseCode(Unknown Source)

[2#6:2] at com.sun.net.ssl.internal.www.protocol.https.HttpsURLConnectionOldImpl.getResponseCode(Unknown Source)

- ¿ Qué ocurre ahora ?

- **Problema: this JSSE implementation is non-pluggable**
- La versión de JSSE del FP NO permite “pluguear” el mismo
- Leed lo siguiente: http://reader.feedshow.com/show_items-feed=2f540a9ea6d902f3f9f1e6eca421c9d7?page=1

[JSSE now fully pluggable](#)

Date: Friday, 02 Dec 2005 08:16

J2SE 5.0 Update 6 was just released yesterday. One change I want to highlight is that **the JSSE framework is now fully pluggable**. This means you can use any 3rd party JSSE provider you wish. No restrictions on the ciphersuites that may be used. No code signing required from the developers that implement a JSSE provider.

So what does this mean for you? Ideally, nothing at all. I say that because I hope you are happy with the SunJSSE provider included in the Sun JDK and that you will just keep using it. But if you do not like SunJSSE, you can choose another provider. Or write your own, it's your choice. And choice is good.

So go ahead and download [5.0u6](#) or a [Mustang](#) build and play. Note the pluggability restrictions were also removed from Mustang a little while ago.

BTW, if you don't like SunJSSE, let us know why so we can fix it. Report a [bug](#), contact the [Java security feedback alias](#) or email me directly and share your pain.

For those of you that care, let me explain a bit of the legal background. *DISCLAIMER: I am not a lawyer. The following is not legal advice, only my limited understanding of the export control rules as they affect JSSE.*

The U.S. government (and many other governments) is concerned that cryptography could be misused by criminals, so it imposes certain restriction on encryption software and hardware that is exported from the U.S. Although those constraints are far less severe than they used to be years ago and from a consumer perspective have all but disappeared, they still cause a bit of trouble for companies like Sun. Apart from lots of paperwork, one big issue that remains is open cryptographic interfaces. In other words, pluggable systems like JCE or JSSE. **Unlimited pluggability is generally not approved.**

That is why at one point we had two versions of JSSE: a domestic version that allowed 3rd party providers and an international one that did not. That was back when JSSE 1.0.x was an optional package for JDK 1.2 and 1.3. When JSSE became part of the core platform in JDK 1.4, maintaining two versions became impractical and we instead only shipped the non-pluggable international version.

Of course we wanted a better solution, and it finally arrived in JDK 5.0. We received approval from the government to allow 3rd party JSSE providers as long as they only supported the ciphersuites on a predefined [list](#). That list includes all currently standardized ciphersuites, so it is a pretty good solution.

Still, the fact that there is any limitation at all was a little annoying. So we tried again and after a some back and forth we received approval to remove that restriction and allow any 3rd party JSSE provider without constraints. This probably does not sound like a bit achievement but it has been a long journey and I am glad it is over now.

Author: "Andreas Sterbenz" Tags: "Java"

- **Problema: this JSSE implementation is non-pluggable**
- La versión **Doméstica de JSSE** Sí es pluggable:
 - <http://java.sun.com/products/archive/jsse/>
 - Refs/jsse-1_0_3_04-domestic.zip
- Incluimos en el servidor HTTP el contenido de:
 - refs\jsse-1_0_3_04-domestic\jsse1.0.3_04\lib\all

Ejercicios Bloque PROT-4

Problema: SSL message with unsupported version ocured

- Con JSSE DOMESTIC bajado remotamente obtenemos lo siguiente cuando llamamos a <https://www.verisign.com> y <https://www.register.com>, lo cual NO ES UN FALLO DE NO TENER PROTOCOLO, es otra cosa....

Verisign:

javax.net.ssl.SSLProtocolException: SSL message with unsupported version ocured

```
at tv.osmosys.security.net.ssl.SSLInputRecord.read(Unknown Source)
at tv.osmosys.security.net.ssl.SSLInputRecord.readProtocolRecord(Unknown Source)
at tv.osmosys.security.net.ssl.SSLSocketImpl.processRecord(Unknown Source)
at tv.osmosys.security.net.ssl.SSLSocketImpl.flushData(Unknown Source)
at tv.osmosys.security.net.ssl.AppDataOutputStream.flush(Unknown Source)
at tv.osmosys.security.net.ssl.SSLSocketImpl.startHandshake(Unknown Source)
at com.sun.net.ssl.internal.www.protocol.https.HttpsClient.doConnect(Unknown Source)
at com.sun.net.ssl.internal.www.protocol.https.NetworkClient.openServer(Unknown Source)
```

Register:

java.io.IOException: unsupported keyword OID.2.5.4.17

```
at com.sun.net.ssl.internal.ssl.AVA.<init>(Unknown Source)
at com.sun.net.ssl.internal.ssl.RDN.<init>(Unknown Source)
at com.sun.net.ssl.internal.ssl.X500Name.a(Unknown Source)
at com.sun.net.ssl.internal.ssl.X500Name.<init>(Unknown Source)
at com.sun.net.ssl.internal.www.protocol.https.HttpsClient.a(Unknown Source)
at com.sun.net.ssl.internal.www.protocol.https.HttpsClient.a(Unknown Source)
at com.sun.net.ssl.internal.www.protocol.https.HttpsClient.a(Unknown Source)
at com.sun.net.ssl.internal.www.protocol.https.HttpURLConnection.connect(Unknown Source)
```

- Comprobemos que los Ciphers que han de soportarse son los apuntados en la tabla referida en *11.8.6 DVB Extensions for Cryptography* **y efectivamente lo son:**

SSL_RSA_WITH_3DES_EDE_CBC_SHA
 SSL_RSA_WITH_DES_CBC_SHA
 SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
 SSL_RSA_WITH_NULL_SHA
 SSL_RSA_WITH_NULL_MD5
 SSL_NULL_WITH_NULL_NULL
 SSL_RSA_WITH_AES_128_CBC_SHA
 SSL_RSA_WITH_AES_256_CBC_SHA

Table 132: Profile of cipher suites that implementations are required to support

CipherSuite	Key Exchange	Cipher	Hash	Value (hex)	MHP status
TLS_NULL_WITH_NULL_NULL (note)	NULL	NULL	NULL	00, 00	Required
TLS_RSA_WITH_NULL_MD5	RSA	NULL	MD5	00, 01	Required
TLS_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1	00, 02	Required
TLS_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA_EXPORT	DES40_CBC	SHA-1	00, 08	Required
TLS_RSA_WITH_DES_CBC_SHA	RSA	DES_CBC	SHA-1	00, 09	Required
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA-1	00, 0A	Required
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	AES	SHA-1	00, 2F	Required
TLS_RSA_WITH_AES_256_CBC_SHA	RSA	AES	SHA-1	00, 35	Required
NOTE: This cipher suite is only used by a TLS implementation during the negotiation of a connection. It is not required to be enabled as a cipher suite that is available for a negotiated connection.					

MHP 1.1.2 A0068r1

Ejercicios Bloque PROT-5

Dos detalles técnicos

- En /logs/"sun.com.net HTTP requests security.txt" se pueden ver las peticiones efectuadas para bajarse el Handler y todo lo que necesita.
- **OJO:** al imprimir la salida con `System.out.println(new String(bytesreaded));` se corta cuando encuentra un acento en la página

```
[2#70:1] P
```

```
[2#70:1] ¡¿°
```

Si en lugar de eso imprimís de la siguiente forma imprime todo:

```
StringBuffer sb = new StringBuffer();  
for (int i =0;i<by.length;i++)  
    sb.append((char)by[i]);  
System.out.println(sb.toString());
```

- La otra forma de especificar los paquetes que habrán de gestionar los diferentes protocolos es la de establecerlos mediante la variable del sistema `java.protocol.handler.pkgs`:

```
System.setProperty("java.protocol.handler.pkgs", "com.sun.net.ssl.internal.www.protocol");  
Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
```

- EL problema es que no sabemos cual, pues **`com.sun.net.ssl.internal.ssl.Provider`** no la tenemos.

¿ Existe otra solución distinta a la de usar `refs/jsse-1_0_3_04-domestic.zip` y establecer el *`URLStreamHandlerFactory`* ?

¿ Qué hay de Apache HTTP Client ?

- **Peticiones HTTPS con Apache HTTPClient**

- Usaremos:

- refs/apache-commons/libs/commons-httpclient-3.1.jar
- refs/apache-commons/libs/commons-logging-1.1.1.jar
- refs/apache-commons/libs/commons-codec-1.3.jar

- Nota: se puede compilar el código eliminando algunas clases de Logging en el paquete `org.apache.commons.logging.impl` al no ser compatibles con PBP 1.1. No afectan siempre que no se use esa capacidad.

- Problemática principal: el tamaño del paquete: 700 kB aprox.
- Solución paliativa: cargar el código usando Priviledged Remote Loading.
- A tener en cuenta siempre: en cualquier caso para acceder a una URL desde una zona de vuestros código siempre será necesario que la aplicación sea “Signed” y disponer de los permisos para acceder al Host deseado.

Ejercicios Bloque PROT-6

- El resultado es positivo, y sin hacer nada más que utilizar el API de Apache. Además se ha resuelto el problema de la petición con Register.com, y puede que el de Verisign no sea tal...

Problemática

- la carga de clases, tanto de com.sun.x o de Apache Commons es un lastre de tiempo que deberemos poder eliminar.
 - Habremos de evaluar la posibilidad de utilizar bien Stored APPS o bien el Persistent Storage para almacenar las clases que necesitamos en este sentido.
 - Otra opción es trabajar usando SSL (si vemos el código de HTTPClient observamos que no hay ningún problema) pero implementando nosotros el protocolo HTTP “a pelo”...
- **¿ y que hay de SSL ?**

Ejercicios Bloque PROT-7

SSL

- Como veis en el ejercicio es factible operar con SSL directamente con Sockets, lo cual tiene sus limitaciones, pero que como ya hemos visto resulta bastante más arduo de acometer (aunque factible).
- Una ventaja importante de usar HTTP/HTTPS es que la gestión de conexiones persistentes es transparente.
- No es objetivo del curso manejar los APIS JSSE en detalle pero sí al menos daros un punto de partida a la vez que demostrar que es posible usar esta funcionalidad con sus complicaciones (quitando la parte de seguridad....no es posible abrir conexiones si la app no está firmada....etc etc)
- En cualquier caso, como ya se ha dicho, el STB ha de soportar por defecto HTTPS, y estamos trabajando con el fabricante para resolver el problema

Para terminar ¿ Donde puedo usar https/https ?

- Prácticamente allí donde puedo usar URLs: 11.11.12 Support for the HTTP protocol in DVB-J
 - In MHP terminals where an HTTP protocol (clause 6.3.7.1 or clause 6.3.7.2) is supported, the following classes and methods shall support the HTTP protocol concerned. In MHP terminals where the HTTPS protocol (clause 6.3.7.3) is supported, the following class and methods shall support that protocol.
 - The constructor for `javax.media.MediaLocator` for referencing audio files intended to be played from memory.
 - Methods on `javax.media.Manager` accepting `javax.media.MediaLocator` as input parameters for constructing JMF players for audio files intended to be played from memory.

AL MENOS EN EL STRONG ESTO NO FUNCIONA

- The classes and methods in the "java.net" package.
- Methods in the present document which accept instances of `java.net.URL` are required to accept instances which encapsulate an "http" URL and behave according to their specification. - e.g. `Toolkit.getImage(java.net.URL)`.

Para terminar ¿ Donde puedo usar https/https ?

- On MHP terminals supporting applications downloaded over the interaction channel as defined in clause 9.7, the method `LocatorFactory.createLocator(String)` shall additionally accept Strings containing URLs using the "http" and "https" protocols as being valid and return a corresponding Locator. This method shall only validate the string to the extent that this is possible without network access.
- On MHP terminals supporting applications downloaded over the interaction channel as defined in clause 9.7, the method `SIManager.getService(Locator)` shall accept such Locators as being valid and return a corresponding `javax.tv.service.Service`. This method shall only validate the locator to the extent that this is possible without network access.
- When HTTP URLs are used with instances of `DVBClassLoader` to load DVB.J classes over the interaction channel in a signed application, the requirements of the MHP security model shall be complied with before a class is allowed to be successfully loaded from such a URL.

ISO/IEC 13818-1	Part 1. Elementary Streams transport definition
ISO/IEC 13818-6	Part 6. Extensions for DSM-CC. Digital Storage Media Command and Control
ETSI EN 300 468	Digital Video Broadcasting (DVB);Specification for Service Information (SI) in DVB systems
ETSI EN 301 192	DVB specification for data broadcasting
ETSI TR 101 202	Implementation Guidelines for Data broadcasting
ETSI TR 101 162	Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems
ETSI TR 102 154	Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in Contribution and Primary Dist
ETSI TR 101 211	Guidelines on implementation and usage of Service Information (SI)
ETSI TR 101 200	Digital Video Broadcasting (DVB); A guideline for the use of DVB specifications and standards
DAVIC	Digital Audio Visual Council. davic 1.4.1
HAVI	Specification of the Home Audio/Video Interoperability (HAVi) Architecture
Interactivetvweb	http://www.interactivetvweb.org/
Wikipedia DSMCC	http://en.wikipedia.org/wiki/DSM-CC
MHP 1.1.2	Multimedia Home Platform, A068r1 & tam668r23_11xdraft_20061115
MHP 1.1.3	Multimedia Home Platform, A068r3
CDC 1.1	Connected Device Configuration (CDC) 1.1 (JSR=218).
PBP 1.1	Personal Basis Profile 1.1 (JSR 217)
MHP.org	www.mhp.org
INTRO MHP 1.1.3	tam1032r1-mhp-iptv-presentation