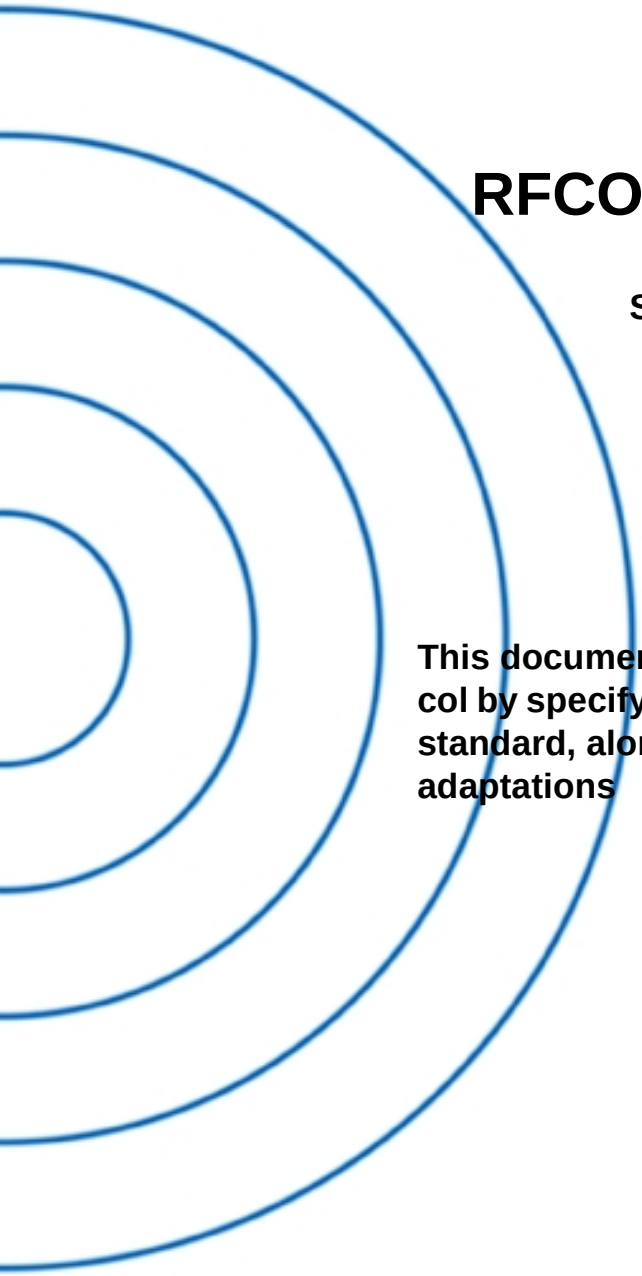
Part F:1



RFCOMM with TS 07.10

Serial Port Emulation

This document specifies the RFCOMM protocol by specifying a subset of the ETSI TS 07.10 standard, along with some Bluetooth-specific adaptations

CONTENTS

1 INTRODUCTION

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. The protocol is based on the ETSI standard TS 07.10. This document does not contain a complete specification. Instead, references are made to the relevant parts of the TS 07.10 standard. Only a subset of the TS 07.10 standard is used, and some adaptations of the protocol are specified in this document.

1.1 OVERVIEW

RFCOMM is a simple transport protocol, with additional provisions for emulating the 9 circuits of RS-232 (EIATIA-232-E) serial ports.

The RFCOMM protocol supports up to 60 simultaneous connections between two BT devices. The number of connections that can be used simultaneously in a BT device is implementation-specific.

1.2 DEVICE TYPES

For the purposes of RFCOMM, a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them. Figure 1.1 shows the complete communication path. (In this context, the term *application* may mean other things than end-user application; e.g. higher layer protocols or other services acting on behalf of end-user applications.)

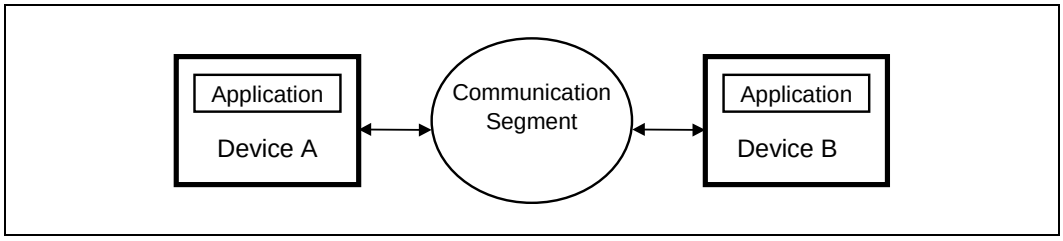


Figure 1.1: RFCOMM Communication Segment

RFCOMM is intended to cover applications that make use of the serial ports of the devices in which they reside. In the simple configuration, the communication segment is a BT link from one device to another (direct connect), see Figure 1.2. Where the communication segment is another network, BT is used for the path between the device and a network connection device like a modem. RFCOMM is only concerned with the connection between the devices in the direct connect case, or between the device and a modem in the network case. RFCOMM can support other configurations, such as modules that communicate via BT on one side and provide a wired interface on the other side, as shown in Figure 1.3. These devices are not really modems but offer a similar service. They are therefore not explicitly discussed here.

Basically two device types exist that RFCOMM must accommodate. Type 1 devices are communication end points such as computers and printers. Type 2 devices are those that are part of the communication segment; e.g. modems. Though RFCOMM does not make a distinction between these two device types in the protocol, accommodating both types of devices impacts the RFCOMM protocol.

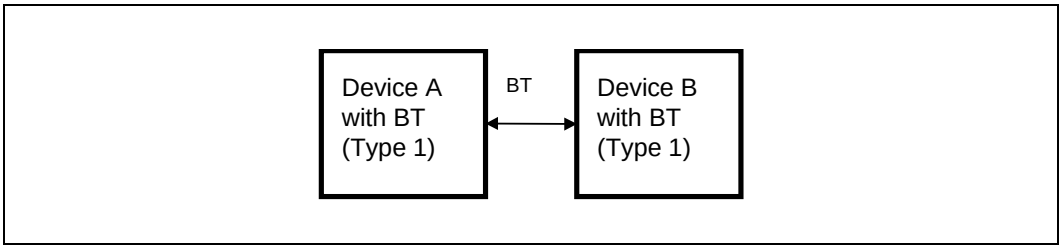


Figure 1.2: RFCOMM Direct Connect

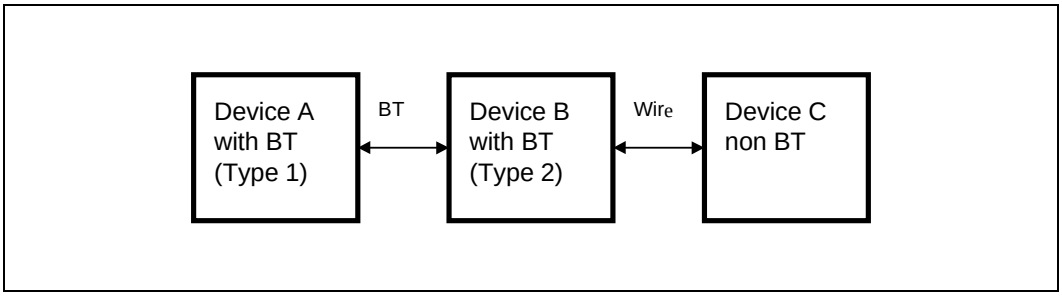


Figure 1.3: RFCOMM used with legacy COMM device

The information transferred between two RFCOMM entities has been defined to support both type 1 and type 2 devices. Some information is only needed by type 2 devices while other information is intended to be used by both. In the protocol, no distinction is made between type 1 and type 2. It is therefore up to the RFCOMM implementers to determine if the information passed in the RFCOMM protocol is of use to the implementation. Since the device is not aware of the type of the other device in the communication path, each must pass on all available information specified by the protocol.

1.3 BYTE ORDERING

This document uses the same byte ordering as the TS 07.10 specification; i.e. all binary numbers are in Least Significant Bit to Most Significant Bit order, reading from left to right.

2 RFCOMM SERVICE OVERVIEW

RFCOMM emulates RS-232 (EIATIA-232-E) serial ports. The emulation includes transfer of the state of the non-data circuits. RFCOMM has a built-in scheme for null modem emulation.

In the event that a baud rate is set for a particular port through the RFCOMM service interface, that will not affect the actual data throughput in RFCOMM; i.e. RFCOMM does not incur artificial rate limitation or pacing. However, if either device is a type 2 device (relays data onto other media), or if data pacing is done above the RFCOMM service interface in either or both ends, actual throughput will, on an average, reflect the baud rate setting.

RFCOMM supports emulation of multiple serial ports between two devices and also emulation of serial ports between multiple devices, see Section 2.3 on page 393.

2.1 RS-232 CONTROL SIGNALS

RFCOMM emulates the 9 circuits of an RS-232 interface. The circuits are listed below.

Pin	Circuit Name
102	Signal Common
103	Transmit Data (TD)
104	Received Data (RD)
105	Request to Send (RTS)
106	Clear to Send (CTS)
107	Data Set Ready (DSR)
108	Data Terminal Ready (DTR)
109	Data Carrier Detect (CD)
125	Ring Indicator (RI)

Table 2.1: Emulated RS-232 circuits in RFCOMM

2.2 NULL MODEM EMULATION

RFCOMM is based on TS 07.10. When it comes to transfer of the states of the non-data circuits, TS 07.10 does not distinguish between DTE and DCE devices. The RS-232 control signals are sent as a number of DTE/DCE independent signals, see Table 2.2.

TS 07.10 Signals	Corresponding RS-232 Control Signals
RTC	DSR, DTR
RTR	RTS, CTS
IC	RI
DV	DCD

Table 2.2: TS 07.10 Serial Port Control Signals

The way in which TS 07.10 transfers the RS-232 control signals creates an implicit null modem when two devices of the same kind are connected together. Figure 2.1 shows the null modem that is created when two DTE are connected via RFCOMM. No single null-modem cable wiring scheme works in all cases; however the null modem scheme provided in RFCOMM should work in most cases.

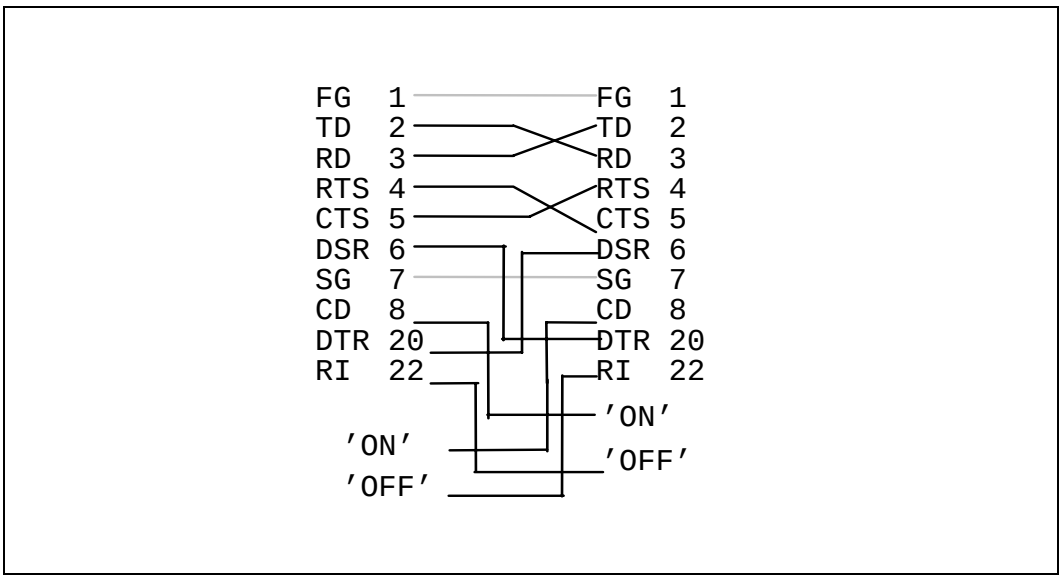


Figure 2.1: RFCOMM DTE–DTE Null Modem Emulation

2.3 MULTIPLE EMULATED SERIAL PORTS

2.3.1 Multiple Emulated Serial Ports between two Devices

Two BT devices using RFCOMM in their communication may open multiple emulated serial ports. RFCOMM supports up to 60 open emulated ports; however the number of ports that can be used in a device is implementation-specific.

A Data Link Connection Identifier (DLCI) [1] identifies an ongoing connection between a client and a server application. The DLCI is represented by 6 bits, but its usable value range is 2…61; in TS 07.10, DLCI 0 is the dedicated control channel, DLCI 1 is unusable due to the concept of Server Channels, and DLCI 62-63 is reserved. The DLCI is unique within one RFCOMM session between two devices. (This is explained further in Section 2.3.2) To account for the fact that both client and server applications may reside on both sides of an RFCOMM session, with clients on either side making connections independent of each other, the DLCI value space is divided between the two communicating devices using the concept of RFCOMM server channels. This is further described in Section 5.4.

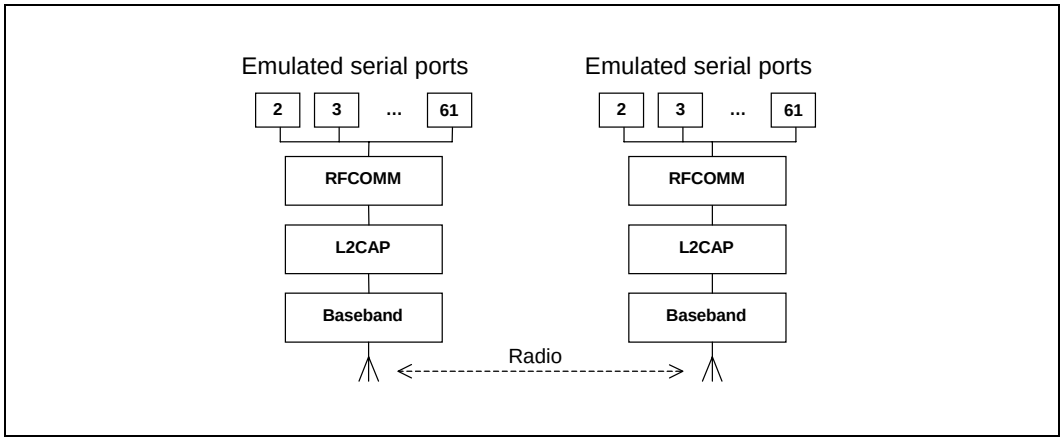


Figure 2.2: Multiple Emulated Serial Ports.

2.3.2 Multiple Emulated Serial Ports and Multiple BT Devices

If a BT device supports multiple emulated serial ports and the connections are allowed to have endpoints in different BT devices, then the RFCOMM entity must be able to run multiple TS 07.10 multiplexer sessions, see Figure 2.3. Note that each multiplexer session is using its own L2CAP channel ID (CID). The ability to run multiple sessions of the TS 07.10 multiplexer is optional for RFCOMM.

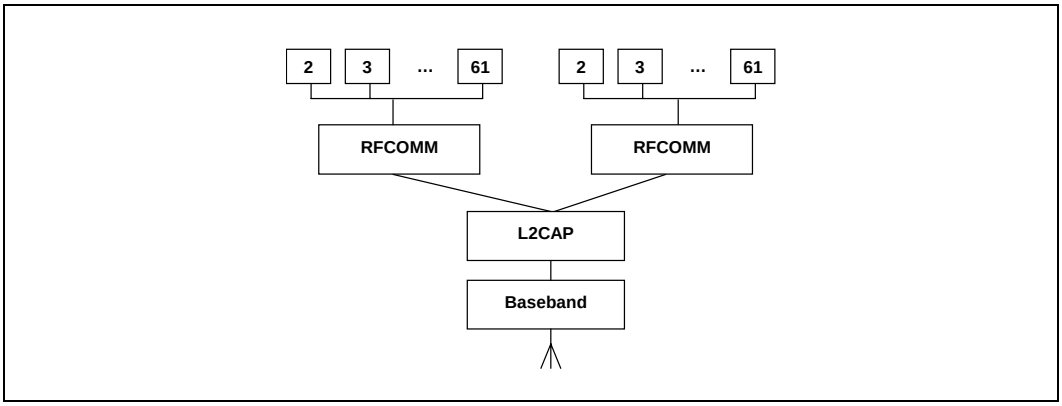


Figure 2.3: Emulating serial ports coming from two BT devices.

3 SERVICE INTERFACE DESCRIPTION

RFCOMM is intended to define a protocol that can be used to emulate serial ports. In most systems, RFCOMM will be part of a port driver which includes a serial port emulation entity.

3.1 SERVICE DEFINITION MODEL

The figure below shows a model of how RFCOMM fits into a typical system. This figure represents the RFCOMM reference model.

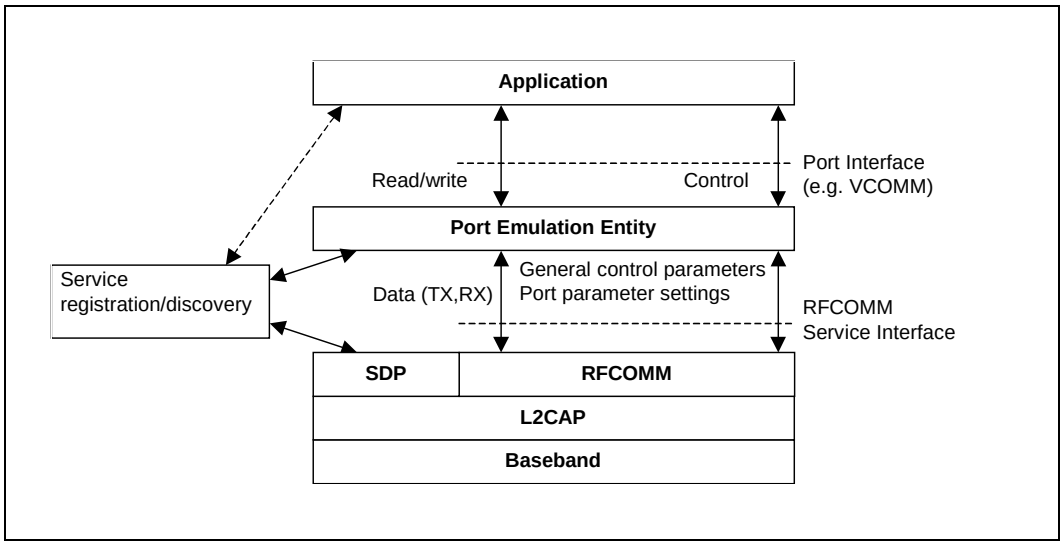


Figure 3.1: RFCOMM reference model

The elements of the RFCOMM reference model are described below.

Element	Description
Application	Applications that utilize a serial port communication interface
Port Emulation Entity	The port emulation entity maps a system-specific communication interface (API) to the RFCOMM services. The port emulation entity plus RFCOMM make up a port driver
RFCOMM	Provides a transparent data stream and control channel over an L2CAP channel. Multiplexes multiple emulated serial ports
Service Registration/ Discovery	Server applications register here on local device, and it provides services for client applications to discover how to reach server applications on other devices
L2CAP	Protocol multiplexing, SAR
Baseband	Baseband protocols defined by BT

4 TS 07.10 SUBSET SUPPORTED BY RFCOMM

4.1 OPTIONS AND MODES

RFCOMM uses the basic option of TS 07.10.

4.2 FRAME TYPES

Table 4.1 shows the TS 7.10 frame types that are supported in RFCOMM.

Frame Types
Set Asynchronous Balanced Mode (SABM) command
Unnumbered Acknowledgement (UA) response
Disconnected Mode (DM) response
Disconnect (DISC) command
Unnumbered information with header check (UIH) command and response

Table 4.1: Supported frame types in RFCOMM

The 'Unnumbered Information (UI) command and response' are not supported by RFCOMM. Since the error recovery mode option of the TS 07.10 protocol is not used in RFCOMM none of the associated frame types are supported.

4.3 COMMANDS

TS 07.10 defines a multiplexer that has a dedicated control channel, DLCI 0. The control channel is used to convey information between two multiplexers. The following commands in TS 07.10 are supported by RFCOMM:

Supported Control Channel Commands
Test Command (Test)
Flow Control On Command (Fcon)
Flow Control Off Command (Fcoff)
Modem Status Command (MSC)
Remote Port Negotiation Command (RPN)
Remote Line Status (RLS)
DLC parameter negotiation (PN)
Non Supported Command Response (NSC)

Whenever a non-supported command type is received a 'Non-Supported Command Response (NSC)' should be sent.

4.4 CONVERGENCE LAYERS

RFCOMM only supports the type 1 convergence layer in TS 07.10.

The Modem Status Command (MSC) shall be used to convey the RS-232 control signals and the break signal for all emulated serial ports.

5 TS 07.10 ADAPTATIONS FOR RFCOMM

5.1 MEDIA ADAPTATION

The opening flag and the closing flags in the 07.10 basic option frame are not used in RFCOMM, instead it is only the fields contained between the flags that are exchanged between the L2CAP layer and RFCOMM layer, see Figure 5.1.

Flag	Address	Control	Length Indicator	Information	FCS	Flag
0111 1101	1 octet	1 octet	1 or 2 octets	Unspecified length but integral number of octets	1 octet	0111 1101

Figure 5.1: Frame Structure for Basic option. Note that the opening and closing flags from the 07.10 Basic option are excluded in RFCOMM.

5.1.1 FCS calculation

In 07.10, the frame check sequence (FCS) is calculated on different sets of fields for different frame types. These are the fields that the FCS are calculated on:

For SABM, DISC, UA, DM frames: on Address, Control and length field.

For UIH frames: on Address and Control field.

(This is stated here for clarification, and to set the standard for RFCOMM; the fields included in FCS calculation have actually changed in version 7.0.0 of TS 07.10, but RFCOMM will not change the FCS calculation scheme from the one above.)

5.2 TS 07.10 MULTIPLEXER START-UP AND CLOSEDOWN PROCEDURE

The start-up and closedown procedures as specified in section 5.7 in TS 07.10 are not supported. This means that the AT-command AT+CMUX is not supported by RFCOMM, neither is the multiplexer close down (CLD) command.

At any time, there must be at most one RFCOMM session between any pair of devices. When establishing a new DLC, the initiating entity must check if there already exists an RFCOMM session with the remote device, and if so, establish the new DLC on that. A session is identified by the Bluetooth BD_ADDR of the two endpoints[1].

5.2.1 Start-up procedure

The device opening up the first emulated serial port connection between two devices is responsible for first establishing the multiplexer control channel. This involves the following steps:

- Establish an L2CAP channel to the peer RFCOMM entity, using L2CAP service primitives, see L2CAP “Service Primitives” on page 295.
- Start the RFCOMM multiplexer by sending SABM command on DLCI 0, and await UA response from peer entity. (Further optional negotiation steps are possible.)

After these steps, DLCs for user data traffic can be established.

5.2.2 Close-down procedure

The device closing the last connection (DLC) on a particular session is responsible for closing the multiplexer by closing the corresponding L2CAP channel.

Closing the multiplexer by first sending a DISC command frame on DLCI 0 is optional, but it is mandatory to respond correctly to a DISC (with UA response).

5.2.3 Link loss handling

If an L2CAP link loss notification is received, the local RFCOMM entity is responsible for sending a connection loss notification to the port emulation/proxy entity for each active DLC. Then all resources associated with the RFCOMM session should be freed.

The appropriate action to take in the port emulation/proxy entity depends on the API on top. For example, for an emulated serial port (vCOMM), it would be suitable to drop CD, DSR and CTS signals (assuming device is a DTE).

1. This implies that, when responding to an L2CAP connection indication, the RFCOMM entity should save and associate the new RFCOMM session with the remote BD_ADDR. This is, at least, necessary if subsequent establishment of a DLC in the opposite direction is possible (which may depend on device capabilities).

5.3 SYSTEM PARAMETERS

Table 5.1 contains all the applicable system parameters for the RFCOMM implementation of the TS 07.10 multiplexer.

System Parameter	Value
Maximum Frame Size *(N1)*	Default: 127 (negotiable range 23 – 32767)
Acknowledgement Timer *(T1)*	60 seconds
Response Timer for Multiplexer Control Channel *(T2)*	60 seconds

Table 5.1: System parameter values

Note: The timer T1 is the timeout for *frames* sent with the P/F-bit set to one (this applies only to SABM and DISC frames in RFCOMM). T2 is the timeout for *commands* sent in UIH frames on DLCI 0.

Since RFCOMM relies on lower layers to provide reliable transmission, the default action performed on timeouts is to close down the multiplexer session. The only exception to this is when trying to set up a new DLC on an existing session; i.e. waiting for the UA response for a SABM command. In this case, the initiating side may defer the timeout by an unspecified amount of time if it has knowledge that the delay is due to user interaction (e.g. authentication procedure in progress). When/if the connection attempt is eventually considered to have timed out, the initiating side must send a DISC command frame on the same DLCI as the original SABM command frame, in order to notify the other party that the connection attempt is aborted. (After that the initiating side will, as usual, expect a UA response for the DISC command.)

5.4 DLCI ALLOCATION WITH RFCOMM SERVER CHANNELS

To account for the fact that both client and server applications may reside on both sides of an RFCOMM session, with clients on either side making connections independent of each other, the DLCI value space is divided between the two communicating devices using the concept of RFCOMM server channels and a direction bit.

The RFCOMM server channel number is a subset of the bits in the DLCI part of the address field in the TS 07.10 frame.

Bit No.	1	2	3	4	5	6	7	8
TS 07.10	EA	C/R	DLCI					
RFCOMM	EA	C/R	D	Server Channel				

Table 5.2: The format of the Address Field

Server applications registering with an RFCOMM service interface are assigned a Server Channel number in the range 1…30. [0 and 31 should not be used since the corresponding DLCIs are reserved in TS 07.10] It is this value that should be registered in the Service Discovery Database, see Section 7.2.

For an RFCOMM session, the initiating device is given the direction bit D=1 (and conversely, D=0 in the other device). When establishing a new data link connection on an existing RFCOMM session, the direction bit is used in conjunction with the Server Channel to determine the DLCI to use to connect to a specific application. This DLCI is thereafter used for all packets in both directions between the endpoints.

In effect, this partitions the DLCI value space such that server applications on the non-initiating device are reachable on DLCIs 2,4,6,…,60; and server applications on the initiating device are reachable on DLCIs 3,5,7,…,61. (Note that for a device that supports multiple simultaneous RFCOMM sessions to two or more devices, the direction bit might not be the same on all sessions.)

An RFCOMM entity making a new DLC on an existing session forms the DLCI by combining the Server Channel for the application on the other device, and the inverse of its own direction bit for the session.

DLCI 1 and 62-63 are reserved and never used in RFCOMM.

5.5 MULTIPLEXER CONTROL COMMANDS

Note that in TS 07.10, some Multiplexer Control commands pertaining to specific DLCIs may be exchanged on the control channel (DLCI 0) *before* the corresponding DLC has been established. (This refers the PN and RPN commands.) All such states associated with an individual DLC must be reset to their default values upon receiving a DISC command frame, or when closing the DLC from the local side. This is to ensure that all DLC (re-)establishments on the same session will have predictable results, irrespective of the session history.

5.5.1 Remote Port Negotiation Command (RPN)

The RPN command can be used before a new DLC is opened and should be used whenever the port settings change.

The RPN command is specified as optional in TS 07.10, but it is mandatory to recognize and respond to it in RFCOMM. (Although the handling of individual settings are implementation-dependent.)

5.5.2 Remote Line Status Command (RLS)

This command is used for indication of remote port line status.

The RLS command is specified as optional in TS 07.10, but it is mandatory to recognize and respond to it in RFCOMM. (Although the handling of individual settings are implementation-dependent.)

5.5.3 DLC parameter negotiation (PN)

The PN command is specified as optional in TS 07.10, but it is mandatory to recognize and respond to it in RFCOMM. This command can be used before a new DLC is opened.

There are some parameters in the PN command which convey information not applicable to RFCOMM. These fields must therefore be set to predetermined values by the sender, and they must be ignored by the receiver. This concern the following fields (see table 3 in ref. [1]):

- I1-I4 must be set to 0. (Meaning: use UIH frames.)
- CL1-CL4 must be set to 0. (Meaning: use convergence layer type 1.)
- T1-T8 must be set to 0. (Meaning: acknowledgment timer *T1*, which is not negotiable in RFCOMM.)
- NA1-NA8 must be set to 0. (Meaning: number of retransmissions *N2*; always 0 for RFCOMM)
- K1-K3 must be set to 0. (Meaning: defines the window size for error recovery mode, which is not used for RFCOMM.)

If a command is received with invalid (or for some reason unacceptable) values in any field, a DLC parameter negotiation response must be issued with values that are acceptable to the responding device.

6 FLOW CONTROL

Wired ports commonly use flow control such as RTS/CTS to control communications. On the other hand, the flow control between RFCOMM and the lower layer L2CAP depends on the service interface supported by the implementation. In addition RFCOMM has its own flow control mechanisms. This section describes the different flow control mechanisms.

6.1 L2CAP FLOW CONTROL IN OVERVIEW

L2CAP relies on the flow control mechanism provided by the Link Manager layer in the baseband. The flow control mechanism between the L2CAP and RFCOMM layers is implementation-specific.

6.2 WIRED SERIAL PORT FLOW CONTROL

Wired Serial ports falls into two camps – software flow control using characters such as XON/XOFF, and flow control using RTS/CTS or DTR/DSR circuits. These methods may be used by both sides of a wired link, or may be used only in one direction.

6.3 RFCOMM FLOW CONTROL

The RFCOMM protocol provides two flow control mechanisms:

1. The RFCOMM protocol contains flow control commands that operate on the aggregate data flow between two RFCOMM entities; i.e. all DLCIs are affected. The control channel commands, FCon and FCoff, are defined in section 5.4.6.3 in ref [1].
2. The Modem Status command as defined in section 5.4.6.3 in ref [1] is the flow control mechanism that operates on individual DLCI.

6.4 PORT EMULATION ENTITY SERIAL FLOW CONTROL

On Type 1 devices some port drivers (Port Emulation Entities plus RFCOMM) will need to provide flow control services as specified by the API they are emulating. An application may request a particular flow control mechanism like XON/XOFF or RTS/CTS and expect the port driver to handle the flow control. On type 2 devices the port driver may need to perform flow control on the non-RFCOMM part of the communication path; i.e. the physical RS-232 port. This flow control is specified via the control parameters sent by the peer RFCOMM entity (usually a type 1 device). The description of flow control in this section is for port drivers on type 1 devices.

Since RFCOMM already has its own flow control mechanism, the port driver does not need to perform flow control using the methods requested by the application. In the ideal case, the application sets a flow control mechanism

and assumes that the COMM system will handle the details. The port driver could then simply ignore the request and rely on RFCOMM's flow control. The application is able to send and receive data, and does not know or care that the port driver did not perform flow control using the mechanism requested. However, in the real world some problems arise.

- The RFCOMM-based port driver is running on top of a packet-based protocol where data may be buffered somewhere in the communication path. Thus, the port driver cannot perform flow control with the same precision as in the wired case.
- The application may decide to apply the flow control mechanism itself in addition to requesting flow control from the port driver.

These problems suggest that the port driver must do some additional work to perform flow control emulation properly. Here are the basic rules for flow control emulation.

- The port driver will not solely rely on the mechanism requested by the application but use a combination of flow control mechanisms.
- The port driver must be aware of the flow control mechanisms requested by the application and behave like the wired case when it sees changes on the non-data circuits (hardware flow control) or flow control characters in the incoming data (software flow control). For example, if XOFF and XON characters would have been stripped in the wired case they must be stripped by the RFCOMM based port driver.
- If the application sets a flow control mechanism via the port driver interface and then proceeds to invoke the mechanism on its own, the port driver must behave in a manner similar to that of the wired case (e.g. If XOFF and XON characters would have been passed through to the wire in the wired case the port driver must also pass these characters).

These basic rules are applied to emulate each of the wired flow control schemes. Note that multiple types of flow control can be set at the same time. Section 5.4.8 in ref [1] defines each flow control mechanism.

7 INTERACTION WITH OTHER ENTITIES

7.1 PORT EMULATION AND PORT PROXY ENTITIES

This section defines how the RFCOMM protocol should be used to emulate serial ports. Figure 7.1 shows the two device types that the RFCOMM protocol supports.

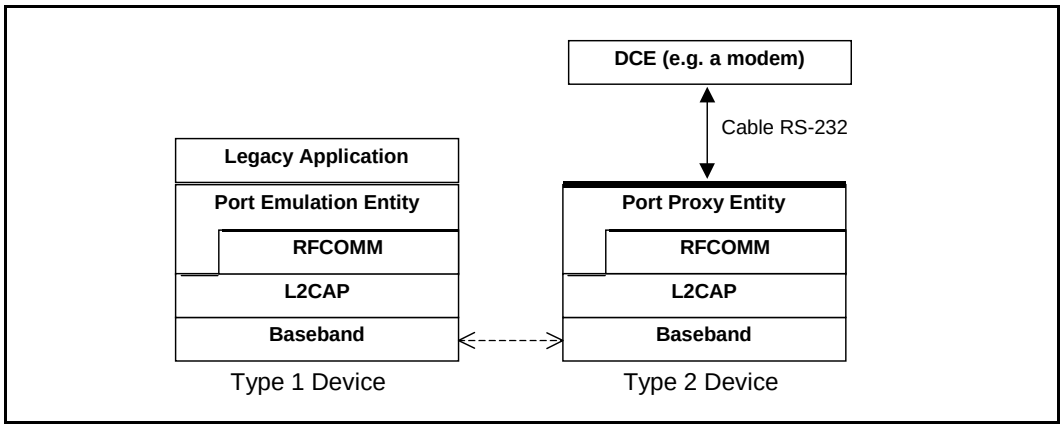


Figure 7.1: The RFCOMM communication model

Type 1 devices are communication endpoints such as computers and printers. Type 2 devices are part of a communication segment; e.g. modems.

7.1.1 Port Emulation Entity

The port emulation entity maps a system specific communication interface (API) to the RFCOMM services.

7.1.2 Port Proxy Entity

The port proxy entity relays data from RFCOMM to an external RS-232 interface linked to a DCE. The communications parameters of the RS-232 interface are set according to received RPN commands, see Section 5.5.1.

7.2 SERVICE REGISTRATION AND DISCOVERY

Registration of individual applications or services, along with the information needed to reach those (i.e. the RFCOMM Server Channel) is the responsibility of each application respectively (or possibly a Bluetooth configuration application acting on behalf of legacy applications not directly aware of Bluetooth).

Below is a template/example for developing service records for a given service or profile using RFCOMM. It illustrates the inclusion of the ServiceClassList with a single service class, a ProtocolDescriptor List with two protocols

(although there may be more protocols on top of RFCOMM). The example shows the use of one other universal attribute (ServiceName). For each service running on top of RFCOMM, appropriate SDP-defined universal attributes and/or service-specific attributes will apply. For additional information on Service Records, see the SDP Specification, Section 2.2 on page 332.

The attributes that a client application needs (at a minimum) to connect to a service on top of RFCOMM are the ServiceClassIDList and the ProtocolDescriptorList (corresponding to the shaded rows in the table below).

Item	Definition	Type/Size	Value	Attribute ID
ServiceClassIDList			Note1	0x0001
ServiceClass0	Note5	UUID/32-bit	Note1	
ProtocolDescriptorList				0x0004
Protocol0	L2CAP	UUID/32-bit	L2CAP /Note1	
Protocol1	RFCOMM	UUID/32-bit	RFCOMM /Note1	
ProtocolSpecificParameter0	Server Channel	Uint8	N = server channel #	
[other protocols]		UUID/32-bit	Note1	
[other protocol-specific parameters]	Note3	Note3	Note3	
ServiceName	Displayable text name	DataElement/ String	’Example service’	Note2
[other universal attributes as appropriate for this service]	Note4	Note4	Note4	Note4
[service-specific attributes]	Note3	Note3	Note3	Note3

Notes:

1. Defined in “Bluetooth Assigned Numbers” on page 1009.
2. For national language support for all ’displayable’ text string attributes, an offset has to be added to the LanguageBaseAttributeIDList value for the selected language (see the SDP Specification, Section 5.1.14 on page 365 for details).
3. To be defined (where necessary) for the specific service.
4. For a specific service some of the SDP-defined universal attributes may apply. See the SDP Specification, Section 5.1 on page 358.
5. This indicates the class of service. It can be a single entry or a list of service classes ranging from generic to most specific.

7.3 LOWER LAYER DEPENDENCIES

7.3.1 Reliability

RFCOMM uses the services of L2CAP to establish L2CAP channels to RFCOMM entities on other devices. An L2CAP channel is used for the RFCOMM/TS 07.10 multiplexer session. On such a channel, the TS 07.10 frames listed in Section 4.2 are sent, with the adaptation defined in Section 5.1.

Some frame types (SABM and DISC) as well as UIH frames with multiplexer control commands sent on DLCI 0 always require a response from the remote entity, so they are acknowledged on the RFCOMM level (but not retransmitted in the absence of acknowledgment, see Section 5.3). Data frames do not require any response in the RFCOMM protocol, and are thus unacknowledged.

Therefore, RFCOMM must require L2CAP to provide channels with maximum reliability, to ensure that all frames are delivered in order, and without duplicates. Should an L2CAP channel fail to provide this, RFCOMM expects a link loss notification, which should be handled by RFCOMM as described in Section 5.2.3.

7.3.2 Low power modes

If all L2CAP channels towards a certain device are idle for a certain amount of time, a decision may be made to put that device in a low power mode (i.e. use hold, sniff or park, see 'Baseband Specification' Section 10.10.3 on page 125). This will be done without any interference from RFCOMM. RFCOMM can state its latency requirements to L2CAP.This information may be used by lower layers to decide which low power mode(s) to use.

The RFCOMM protocol as such does not suffer from latency delays incurred by low power modes, and consequentially, this specification does not state any maximum latency requirement on RFCOMM's behalf. Latency sensitivity inherently depends on application requirements, which suggests that an RFCOMM service interface implementation could include a way for applications to state latency requirements, to be aggregated and conveyed to L2CAP by the RFCOMM implementation. (That is if such procedures make sense for a particular platform.)

8 REFERENCES

[1] TS 07.10, ver 6.3.0, ETSI

[2] Bluetooth L2CAP Specification

[3] Bluetooth SDP Specification

[4] Bluetooth Assigned Numbers

9 TERMS AND ABBREVIATIONS

The following terms are used throughout the document.

DTE	Data Terminal Equipment – in serial communications, DTE refers to a device at the endpoint of the communications path; typically a computer or terminal
DCE	Data Circuit-Terminating Equipment – in serial communications, DCE refers to a device between the communication endpoints whose sole task is to facilitate the communications process; typically a modem
RFCOMM initiator	The device initiating the RFCOMM session; i.e.setting up RFCOMM channel on L2CAP and starting RFCOMM multiplexing with the SABM command frame on DLCI 0 (zero)
RFCOMM Client	An RFCOMM client is an application that requests a connection to another application (RFCOMM server)
RFCOMM Server	An RFCOMM server is an application that awaits a connection from an RFCOMM client on another device. What happens after such a connection is established is not within the scope of this definition
RFCOMM Server Channel	This is a subfield of the TS 07.10 DLCI number. This abstraction is used to allow both server and client applications to reside on both sides of an RFCOMM session