

C++ in Telecommunication – Project UE

Specification

Author	Piotr Nycz, ...
--------	-----------------

This material, including documentation and any related computer programs, is protected by copyright controlled by Nokia Solutions and Networks. All rights are reserved. Copying, including reproducing, storing, adapting or translating, any or all of this material requires the prior written consent of Nokia Solutions and Networks. This material also contains confidential information, which may not be disclosed to others without the prior written consent of Nokia Solutions and Networks.

Contents

1	Basic information	6
1.1	Project elements.....	6
1.2	Communication scheme	6
1.3	BTS	7
1.4	UE	8
1.5	COMMON.....	8
2	UE basic information	8
2.1	Actors	9
2.1.1	UE	9
2.1.2	User.....	9
2.1.3	BTS	9
2.1.4	SMS DB	9
2.1.5	Timer	10
2.2	States	10
3	UE messages from/to BTS	11
3.1	Constructing and reading messages.....	11
3.1.1	Field types	12
3.1.2	Bytes order	12
3.2	UE ⇔ BTS messages	12
3.2.1	SIB message	12
3.2.2	Attach messages	12
3.3	UE ⇔ BTS ⇔ UE messages.....	13
3.3.1	From/To.....	13
3.3.2	Encryption	13
3.3.3	SMS message	15
3.3.4	Call messages	16
3.4	BTS → UE errors indication messages	16

3.4.1	Unknown Recipient.....	16
3.4.2	Unknown Sender	17
4	UE features (to implement)	17
4.1	UE basic features	18
4.1.1	Attach to BTS	18
4.1.2	Re-Attach to BTS.....	21
4.1.3	Receiving SMS.....	22
4.1.4	Viewing SMS	23
4.1.5	Sending SMS	24
4.1.6	Receiving Call Request	27
4.1.7	Sending Call Request.....	32
4.1.8	Dropping Call.....	39
4.1.9	Call (talking)	41
4.2	UE basic interactions	46
4.2.1	Interaction with Attach to BTS procedure.....	46
4.2.2	Interactions with Re-Attach	49
4.2.3	Interactions with Receiving SMS	50
4.2.4	Interactions with Viewing SMS.....	50
4.2.5	Interactions with Sending SMS	51
4.2.6	Interactions with Receiving Call Request	52
4.2.7	Interactions with Sending Call Request	52
4.2.8	Interactions with Dropping Call	54
4.2.9	Interactions with Call (Talking).....	54
4.3	UE advanced features	54
5	List of figures	56
6	List of tables	57
7	List of acronyms	57

1 Basic information

1.1 Project elements

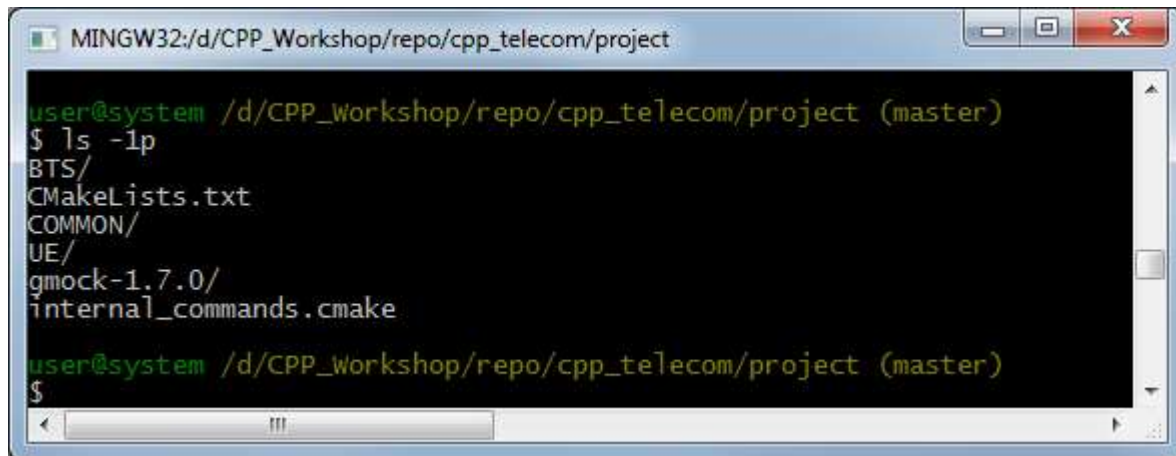
One can view telecommunication as a network of communicating elements.

Project described in this specification consists of two network elements (NE), where naming is taken from LTE/3GPP standards:

- BTS – Base Transceiver Station
- UE – User Equipment (like mobile phone)

Besides of NEs, there is one common to BTS and UE software library:

- COMMON – BTS/UE Common library



```
MINGW32:/d/CPP_Workshop/repo/cpp_telecom/project
user@system /d/CPP_Workshop/repo/cpp_telecom/project (master)
$ ls -lp
BTS/
CMakeLists.txt
COMMON/
UE/
gmock-1.7.0/
internal_commands.cmake
user@system /d/CPP_Workshop/repo/cpp_telecom/project (master)
$
```

Figure 1 Project directory structure

1.2 Communication scheme

The communication scheme between NEs is as follows:

- UE communicates with only one BTS.
- BTS communicates with many UEs.
- UEs communicate to each other only via BTS.
- BTS do not communicate to each other (at least in the current phase of project)

There is no limitation on *how many NEs can be deployed on how many computers* other than types used to represent UE ID (**PhoneNumber**) or BTS ID (**BtsId**). Both types are defined by COMMON library.

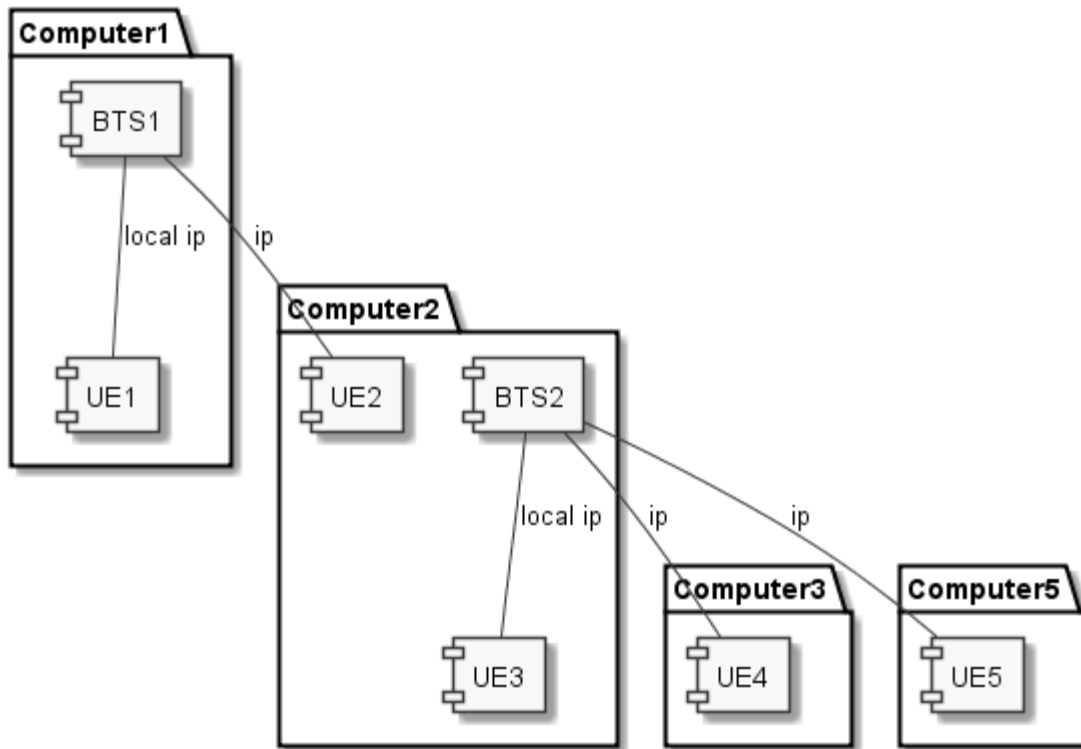


Figure 2 Possible deployment

Looking at [Figure 2 Possible deployment](#) the following communication are possible:

- UE1 and BTS1
- UE1 and UE2 (via BTS1)
- UE2 and BTS1
- UE3 and BTS2
- UE3 and UE4 (via BTS2)
- UE3 and UE5 (via BTS2)
- U4 and BTS2
- UE4 and UE5 (via BTS2)
- UE5 and BTS2

1.3 BTS

BTS is fully implemented and it is given as it is.

It is not student task to change or extend BTS functionalities.

1.4 UE

Student task is to implement UE as it is specified in subsequent chapters of this specification.

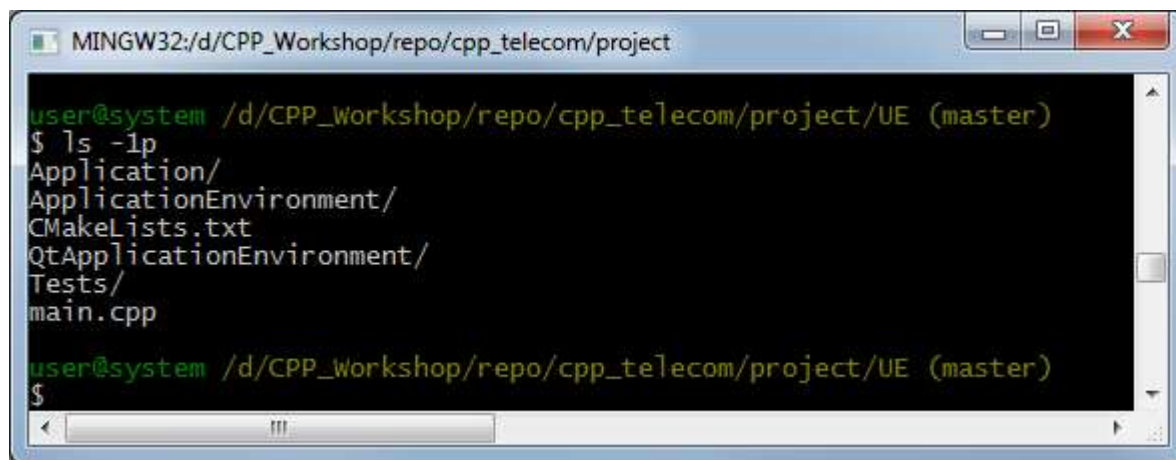
1.5 COMMON

Student can use COMMON library in a way as it is used by BTS.

Student can extend COMMON library; but student shall not remove or change this library in a way that changes BTS behavior.

The COMMON library unit tests and interface documentations present in header files are available to help understand how this library can be used.

2 UE basic information



```

MINGW32:/d/CPP_Workshop/repo/cpp_telecom/project

user@system /d/CPP_Workshop/repo/cpp_telecom/project/UE (master)
$ ls -lp
Application/
ApplicationEnvironment/
CMakeLists.txt
QtApplicationEnvironment/
Tests/
main.cpp

user@system /d/CPP_Workshop/repo/cpp_telecom/project/UE (master)
$
    
```

Figure 3 UE directory structure

UE is left partially unimplemented:

- To be implemented:
 - Application code (Application directory)
 - main.cpp: it already starts ApplicationEnvironment, but using of Application is missing.
- Already implemented:
 - ApplicationEnvironment: interface library

- interface documentation is present in header files to help understand how to use this interface library.
- QtApplicationEnvironment: implementation library
 - It is just one of possible implementation of ApplicationEnvironment

Student shall be aware that one of possible checks of their project is to test if it works with other implementation of ApplicationEnvironment, other than QtApplicationEnvironment!

It is important:

- Not to use Qt library in Application directly
- Not to add anything new to ApplicationEnvironment

2.1 Actors

In subsequent chapters UML Sequence or Use Case Diagrams are used to describe requirements. In current chapter – list of actors used in these diagrams are briefly described.

2.1.1 UE

The UE application to be written by students.

In UML Diagrams, UE represents only Application part of UE.

Moreover, some parts of this application are represented by some other actors, like [SMS DB](#) or [Timer](#).

2.1.2 User

Real UE user, but seen from UE Application POV as `ue::IUeGui` interface from ApplicationEnvironment.

The User actor actions are not just functions from `ue::IUeGui`, but rather some descriptions of real UE User actions.

2.1.3 BTS

BTS NE, but seen from UE Application POV as `ue::ITransport` interface from ApplicationEnvironment.

The BTS actor actions are just simplified description of real messages that can be sent between UE and BTS, or between UE and other UE via BTS.

2.1.4 SMS DB

Part of [UE](#) Application.

Responsibilities of this actor are to represent operations on SMS repository within [UE](#) Application.

SMS DB is an Actor, but this does not mean, that it shall be implemented as single object or single module. It can be as well: set of objects, or part of some bigger object.

2.1.5 Timer

Part of [UE](#) Application.

Responsibilities of this actor is to represent operations on timers' machinery within [UE](#) Application.

Operations:

- start (UE→Timer)
- cancel (UE→Timer)
- timeout (Timer→UE)

It is student decision if UE Application implementation will require one or many timers being active at the same time.

Timer is an Actor, but this does not mean, that it shall be implemented as single object or single module. It can be as well: set of objects, or part of some bigger object.

2.2 States

In subsequent chapters UML Sequence or Use Case Diagrams are used to describe requirements. In current chapter – states and their transitions are presented in [Figure 4 UE Application States](#).

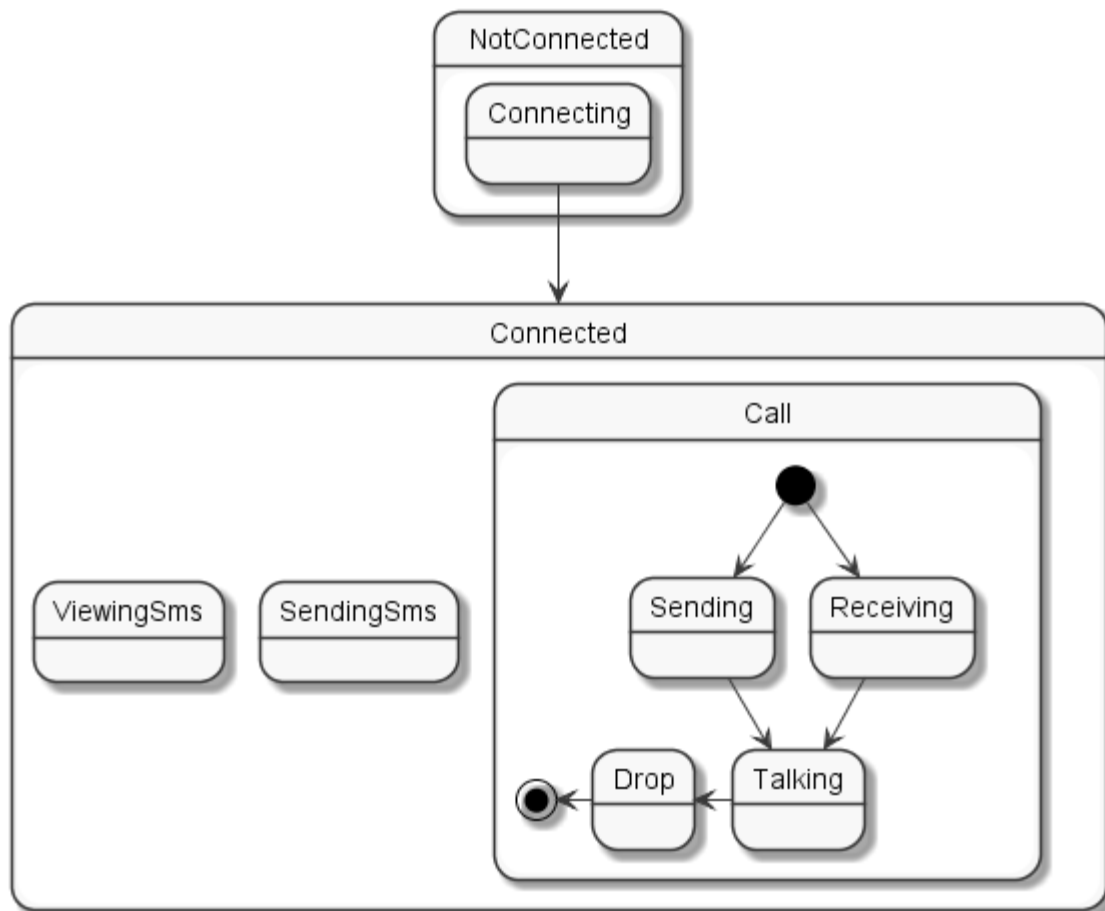


Figure 4 UE Application States

3 UE messages from/to BTS

3.1 Constructing and reading messages

Messages are to be sent and received as a stream of bytes.

Messages shall be constructed and read by using of COMMON library:

- `class IncomingMessage` for reading message

- `class OutgoingMessage` for constructing message

There is no padding or special alignment between fields in messages, so student cannot just define C++ `struct` type “representing” message and cast bytes stream to it. Moreover, endianness format of received messages might differ from your CPU endianness format.

3.1.1 Field types

All types used to describe messages (like `MessageId` or `PhoneNumber`) are defined in [COMMON](#) library.

3.1.2 Bytes order

Bytes in messages are and should be ordered according to big endian format, i.e. in two bytes word, the most significant byte is first. Big endian format is the network standard.

Be aware that architecture of some processors are not big endian format (e.g. Intel processors are little endian format – the least significant byte is first).

Fortunately, already mentioned COMMON classes will do all necessary conversion between endianness formats in the background, no special action needed.

3.2 UE ↔ BTS messages

3.2.1 SIB message

Direction		BTS→UE		
Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::Sib	
	from	PhoneNumber		value not used
	to	PhoneNumber		value not used
btsId		BtsId	Sender ID	

Table 1 SIB (System Information Broadcast) message

3.2.2 Attach messages

Direction		UE→BTS		
Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::AttachRequest	
	from	PhoneNumber	UE Number	
	to	PhoneNumber		value not used
btsId		BtsId	Receiver ID	

Table 2 Attach Request

Direction		BTS→UE		
Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::AttachResponse	
	from	PhoneNumber		value not used
	to	PhoneNumber	UE Number	
accepted		bool (implemented as std::uint_8)		accepted(true) rejected(false)

Table 3 Attach Response

3.3 UE ↔ BTS ↔ UE messages

3.3.1 From/To

All the UE ↔ UE messages can be received on UE or sent from UE. Depending on it – Message header fields From/To mean:

Direction		UE→BTS
Field		Meaning
Header	from	UE own Number
	to	Receiver UE number

Table 4 Message header from/to fields meaning when message is sent from UE

3.3.2 Encryption

Method of encryption can be specified in free form as sequence of bytes – as long as both UE understand it.

The predefined methods of encryption:

1. “no encryption” represented as one byte containing value of zero.
2. “XOR encryption” represented as one byte containing value of one.
3. “Cesar encryption” represented as one byte containing value of two.
4. “RSA encryption” represented as one byte containing value of three.

Field	Type	Value	Comment
Mode	<code>std::uint8_t</code>	Differ from 0	
Data	<code>std::uint8_t[N]</code>	Unspecified	N is unspecified – might be zero. Interpretation depends on mode value.

Table 5 Encryption for encrypted communication

3.3.2.1 No encryption

Field	Type	Value	Comment
mode	<code>std::uint8_t</code>	0	N=0 (no data)

Table 6 Encryption for not encrypted communication

3.3.2.2 XOR encryption

Every byte is XORed with key.

Field	Type	Value	Comment
Mode	<code>std::uint8_t</code>	1	
Data	<code>std::uint8_t</code>	0-255	Encryption “key”

Table 5.1 Encryption for XOR encrypted communication

Encryption “key” shall be used to encrypt Text field of subsequent Call Talk or SMS messages.

Be aware that two UEs may use different values as their encryption “key”.

3.3.2.3 Cesar encryption

Every byte is encoded as <original value> + <key>.

Field	Type	Value	Comment
Mode	std::uint8_t	2	
Data	std::uint8_t	0-255	Encryption “key”

Table 6.2 Encryption for Cesar encrypted communication

Encryption “key” shall be used to encrypt Text field of subsequent Call Talk or SMS messages.

Be aware that two UEs may use different values as their encryption “key”.

3.3.2.4 RSA encryption

Entire message is encrypted with a RSA private or public key.

Field	Type	Value	Comment
Mode	std::uint8_t	3	
KeyLength	std::uint16_t	>0	Length of public key
Data	std::uint8_t[KeyLength]	0-255	Public RSA key in PEM format

Table 7.3 Encryption for RSA encrypted communication

RSA encryption/decryption rules for Call Talk messages are different from those for SMS messages.

MessageId	Encryption method	Decryption method
SMS	Encrypt with own private key	Decrypt with peer public key
CallTalk	Encrypt with peer public key	Decrypt with own private key

3.3.3 SMS message

Field	Type	Value	Comment
Header	messageId	MessageId	MessageId::Sms
	from	PhoneNumber	See From/To
	to	PhoneNumber	
encryption			See Encryption
text	char[N]	SMS text	Size is just to the end of Message

Table 8 SMS

3.3.4 Call messages

Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::CallRequest	
	from	PhoneNumber		See From/To
	to	PhoneNumber		
encryption			See Encryption	See Encryption

Table 9 Call Request

Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::CallAccepted	
	from	PhoneNumber		See From/To
	to	PhoneNumber		
encryption			See Encryption	

Table 10 Call Accepted

Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::CallDropped	
	from	PhoneNumber		See From/To
	to	PhoneNumber		

Table 11 Call Dropped

Field		Type	Value	Comment
Header	messageld	MessageId	MessageId::CallTalk	
	from	PhoneNumber		See From/To
	to	PhoneNumber		
text		char[N]	Call text	Size is just to the end of Message

Table 12 Call Talk

3.4 BTS → UE errors indication messages

3.4.1 Unknown Recipient

Field		Type	Value	Comment
Header	messageId	MessageId	MessageId::UnknownRecipient	
	from	PhoneNumber		value not used
	to	PhoneNumber	UE Number	
Header of failing message	messageId	MessageId	Copied from failed message	
	from	PhoneNumber		
	to	PhoneNumber		

Table 13 Unknown Recipient

The [Unknown Recipient](#) message indicates that BTS does not know the recipient, the recipient UE is unknown.

This is sometimes expected case, when, e.g., UE sends SMS to peer UE and that UE is not connected to BTS.

3.4.2 Unknown Sender

Field		Type	Value	Comment
Header	messageId	MessageId	MessageId::UnknownSender	
	from	PhoneNumber		value not used
	to	PhoneNumber	UE Number	
Header of failing message	messageId	MessageId	Copied from failed message	
	from	PhoneNumber		
	to	PhoneNumber		

Table 14 Unknown Sender

The [Unknown Sender](#) message shall not be received, it indicates implementation (logic) error.

The only requirement is to log this message (when received) as error in ApplicationEnvironment/Logger object.

4 UE features (to implement)

This chapter defines UE behaviour.

This behaviour is divided into list of features in subsequent subchapters.

For any missing features or requirements – if it appears that some UE behaviour is not defined by this chapter – student is free to implement it in “own” way.

All requirements specified as either UML Use Case or UML Sequence Diagrams shall be tested.

See [UE messages from/to BTS](#) for messages definitions.

See [UE basic information](#) for actors and states definition.

4.1 UE basic features

4.1.1 Attach to BTS

UE has to perform special attach procedure to connect to BTS.

Main goal is to inform BTS about UE phone number and to inform BTS that UE exists.

Only after being connected, UE can perform other procedures, like receiving/sending SMS.

4.1.1.1 Attach to BTS – Use Case

Preconditions:

- UE in **NotConnected** state:
 - [main] BTS is on, UE is starting;
 - [alternate] UE was already started, when BTS starts
 - [alternate] Previous attempt(s) to attach failed
- TCP/IP connection exist (or BTS and UE on the same host)
 - UE has configured (server,port) proper connection to BTS
 - Default is server=localhost, port=8181 – and this is ok for localhost

Main success scenario:

1. BTS sends **SIB(btsId)** message to UE
2. UE sends **AttachRequest(itsPhoneNumber)** to BTS
3. BTS responds with **AttachResponse(uePhoneNumber, accepted:true)**
4. UE signals “I am connected” to the User

Main failure scenario:

- 3a. BTS responds with **AttachResponse(uePhoneNumber, accepted:false)**

Alternate failure scenario:

- 3b. BTS does not respond within 500ms timeout

Postconditions:

- [success] UE is ready for serving User: UE is in **Connected** state.
- [failure] UE is not ready for serving Use: UE is in **NotConnected** state.

4.1.1.2 Attach to BTS Sequence Diagram – Success

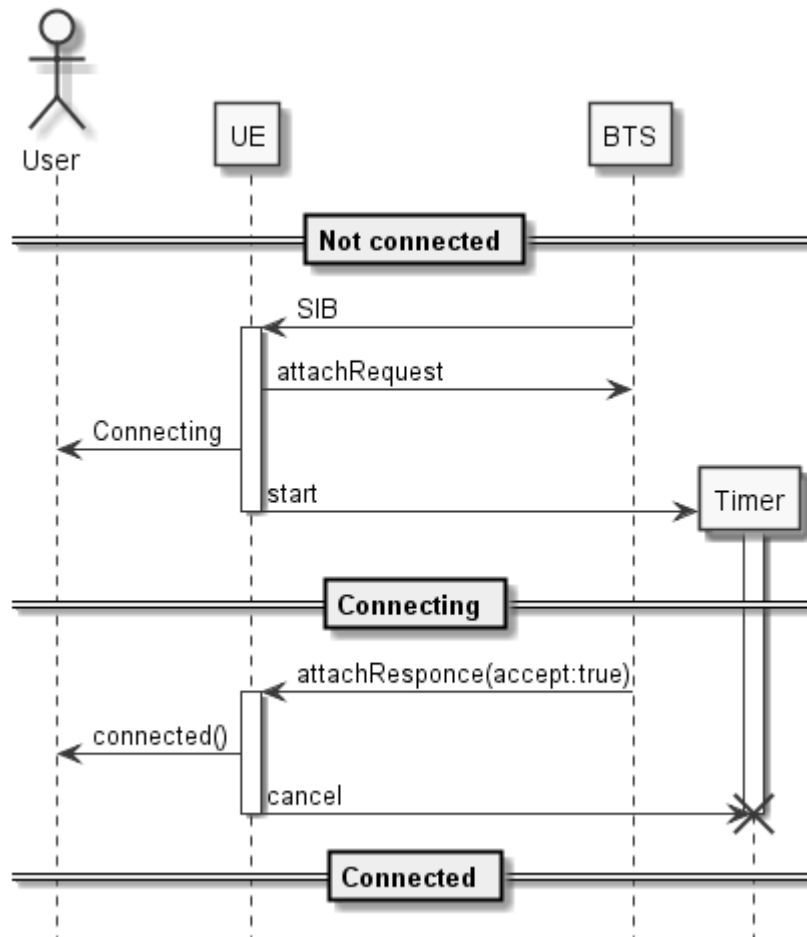


Figure 5 Attach to BTS - Success

4.1.1.3 Attach to BTS Sequence Diagram – Failure: BTS rejected

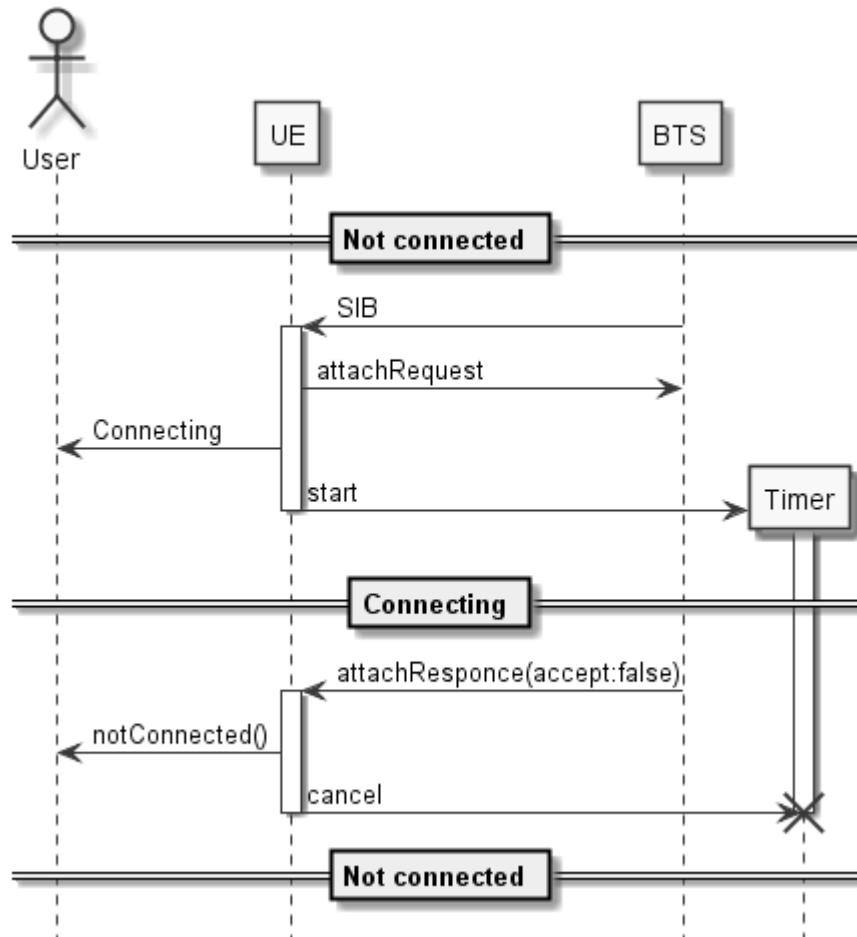


Figure 6 Attach to BTS, BTS rejected

4.1.1.4 Attach to BTS Sequence Diagram – Failure: timeout

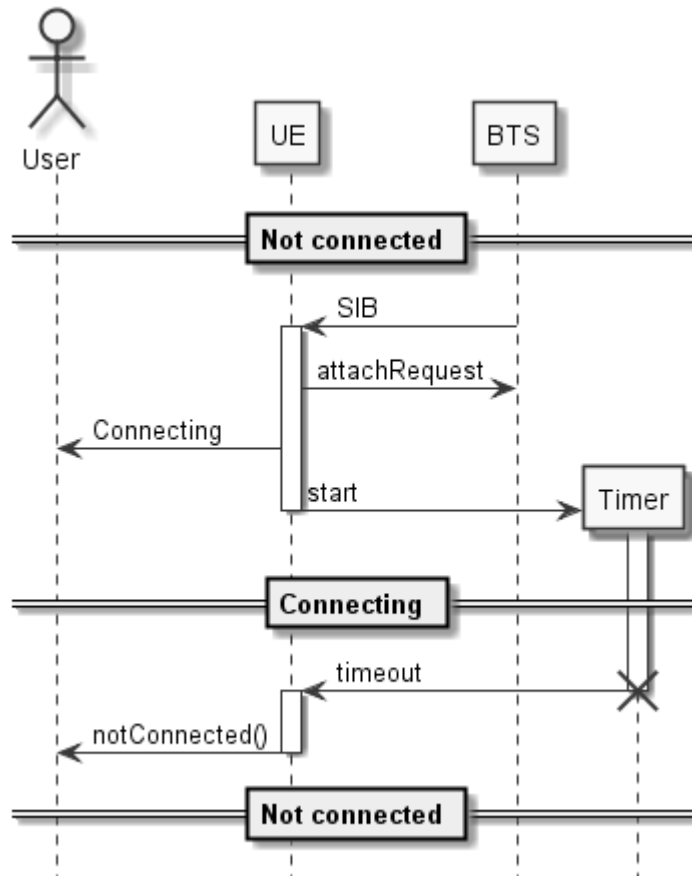


Figure 7 Attach to BTS, timeout

4.1.2 Re-Attach to BTS

UE has to be able to perform subsequent attach procedure, if disconnected. This procedure consists of two sub-procedures:

- 1) Disconnecting
- 2) Attaching

Attaching while reattaching does not differ from initial attaching described in 4.1.1 Attach to BTS.

Disconnecting is information received from Transport object that connection to BTS was drop.

Transport object is part of UE ApplicationEnvironment (see [UE basic information](#)) – for details refer to Transport object interface definition - see [UE basic information](#) to find header file location.

4.1.2.1 Disconnecting – Use Case

Preconditions:

- UE in **Connecting** or **Connected**:

Scenario:

1. **ApplicationEnvironment/Transport** informs UE via registered callback that connection to BTS was drop

Postconditions:

- UE is in **NotConnected** state.
- UE is ready to received next Sib message to start new Attach procedure

4.1.3 Receiving SMS

UE can receive SMS from any other UE connected to the same BTS.
UE has to store these SMS to allow its User to view them.

4.1.3.1 Receiving SMS – Use Case Scenario

Preconditions:

- UE in **Connected** state:

Scenario:

1. UE receives **SMS(from,text)** message from BTS
2. UE stores it in its SMS DB (as SMS not yet read)
3. UE informs User that there is new SMS.

Postconditions:

- Received **SMS(from,text)** stored in SMS DB
- User is informed new SMS arrived

4.1.3.2 Receiving SMS Sequence Diagram

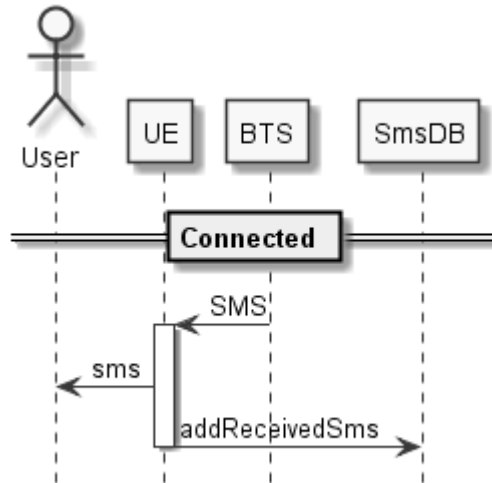


Figure 8 Receiving SMS

4.1.4 Viewing SMS

User shall be able to view SMS on UE.

UE shall maintain “read/not read” property of received SMS.

4.1.4.1 Viewing SMS – Use Case

Preconditions:

- UE is presenting main menu actions to the User

Main success scenario:

1. User selects on UE: view SMS action (might be implement as menu/selection item)
2. UE shows SMS List to User (might be implemented as selection items)
3. User selects one SMS
4. UE shows selected SMS to User
 - 4.1 UE for “not read” SMS changes this SMS property to “read”
5. User closes the SMS view
6. User closes the SMS List view

Alternative success scenario

- 6a. User selects another SMS to view – i.e. go to p4

Alternative success scenario

- 2a SMS List is empty – i.e. go to p6

Postconditions:

- All viewed SMS marked as “read” in SMS DB
- UE is presenting main menu actions to the User

4.1.4.2 Viewing SMS Sequence Diagram – Success

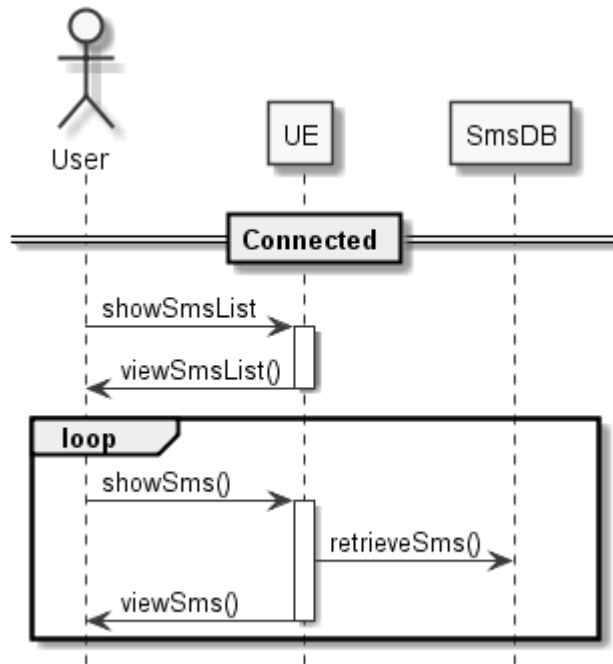


Figure 9 Viewing SMS

4.1.5 Sending SMS

UE, when connected, shall be able to allow User to send SMS to other UE (via BTS).

4.1.5.1 Sending SMS – Use Case

Preconditions:

- UE in **Connected** state

Success scenario:

1. User selects on UE: send SMS action (might be implement as menu/selection item)
2. UE switches to edit SMS mode
3. User types other UE Phone Number and text of SMS
4. User clicks Accept (green “pick up handset” button)
5. UE sends to BTS **SMS(from=myPhoneNumber,to=enteredPhoneNumber,text=enteredText)**
6. UE stores SMS as sent SMS

7. UE switches to main menu mode

Failure scenario

3a. User clicks Reject (red “hang up the phone” button)

Alternative failure scenario

4a. User clicks Reject (red “hang up the phone” button)

(Optional) Alternative partial-success scenario

8a. [Unknown Recipient](#) message is received with body indicating that SMS sent in p6 was not received. SMS might be marked in SMS DB as sent but not received.

Postconditions

- UE in [Connected](#) state
- [success] UE sends SMS and stores it in SMS DB

4.1.5.2 Sending SMS Sequence Diagram – Success

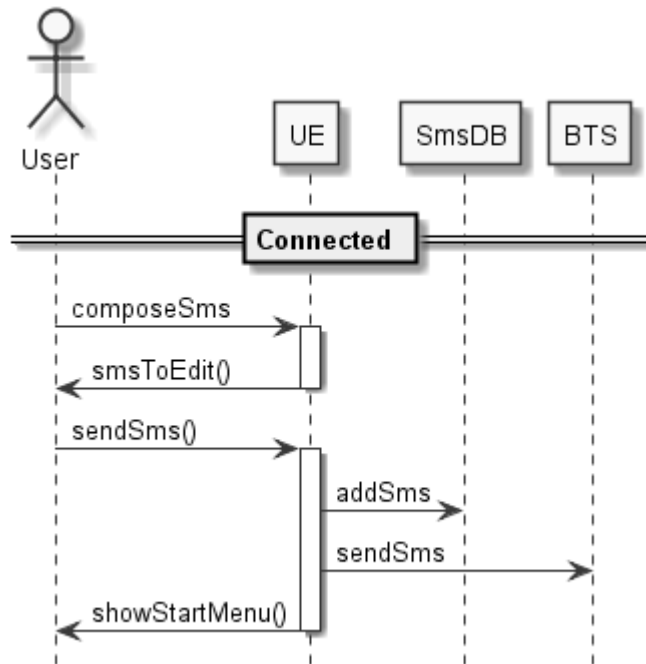


Figure 10 Sending SMS – Success

4.1.5.3 Sending SMS Sequence Diagram – SMS sent to unknown recipient

This sequence diagram is one of possible many solution to the case when UE application receives [Unknown Recipient](#) message in response to SMS. It means, that this specification does not force UE Application implementation to behave in this way presented here:

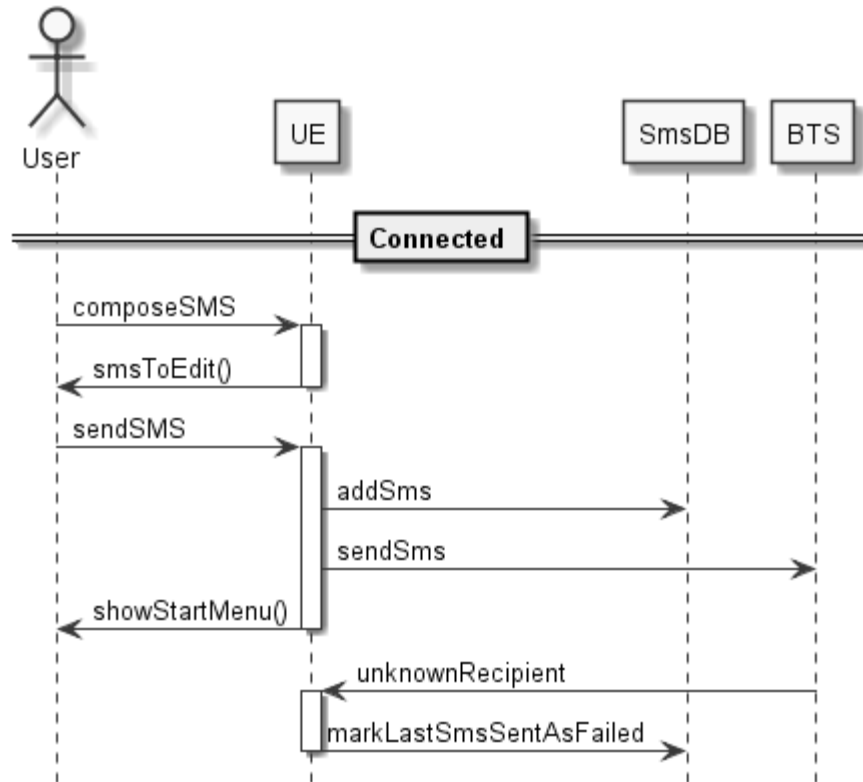


Figure 11 Sending SMS – Partial Success

4.1.5.4 Sending SMS Sequence Diagram – Failure (Abort)

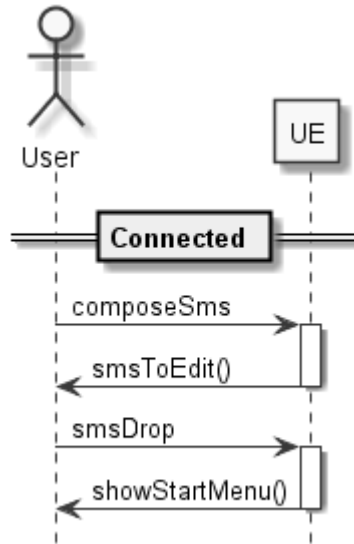


Figure 12 Sending SMS - Abort

4.1.6 Receiving Call Request

UE shall present to User the incoming call Request from other UE (via BTS).

User shall see incoming phone number and two options:

1. Accept call – thus starting conversation
2. Reject call

According to the User action – proper response message shall be sent to other UE (via BTS).

UE shall guard User reaction with timeout, i.e. when User does not react in the specified time (see proper sequence diagram) then User Reject action is assumed.

This is mirror procedure to [Sending Call Request](#).

4.1.6.1 Receiving Call Request – Use Case

Preconditions:

- UE in **Connected** state:

Main success scenario:

1. BTS sends **CallRequest(from=callingPhoneNumber)** message to UE
2. UE informs User about incoming call
3. User selects Accept option (green “pick up handset” button)
4. UE changes its mode to conversation mode (**Talking** state)

5. UE sends to BTS `CallAccept(to=callingPhoneNumber)`

Main failure scenario – Reject

3a. User selects Reject option (red “hang up the phone” button)

4a. UE returns to main menu mode

5a. UE sends to BTS `CallDropped(to=callingPhoneNumber)`

Alternate failure scenario - Timeout

3a. User does not perform any action within 30s timeout

4a. UE returns to main menu mode

5a. UE sends to BTS `CallDropped(to=callingPhoneNumber)`

Alternate failure scenario – Peer UE becomes unknown after CallAccepted

6. BTS sends `UnknownRecipient(failedHeader.to=callingPhoneNumber)`

7. Inform UE User that Peer User was disconnected from BTS

- UE GUI AlertMode can be used for this (for a short moment display some message)

8. UE behaves as if Peer User dropped call

- see [Dropping Call Sequence Diagram – Initiated by other User](#)

Alternate failure scenario – Peer UE becomes unknown after CallDropped

6a. BTS sends `UnknownRecipient(failedHeader.to=callingPhoneNumber)`

7a. Message is ignored.

Postconditions

- [success] UE is ready for talking (`Talking` state)
- [failure] UE returns to `Connected` state.

4.1.6.2 Receiving Call Request Sequence Diagram – Success (Accept)

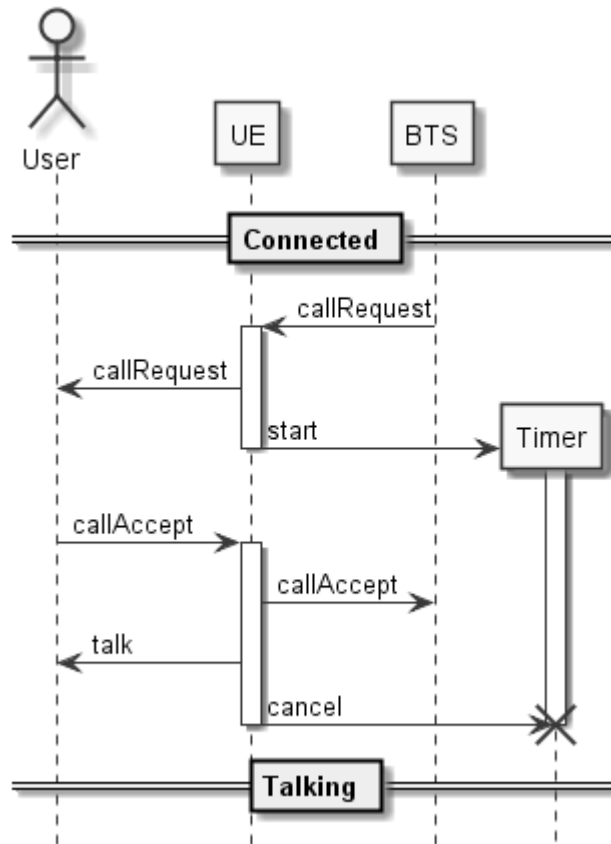


Figure 13 Received Call Request Accepted

4.1.6.3 Receiving Call Request Sequence Diagram – Failure (Reject)

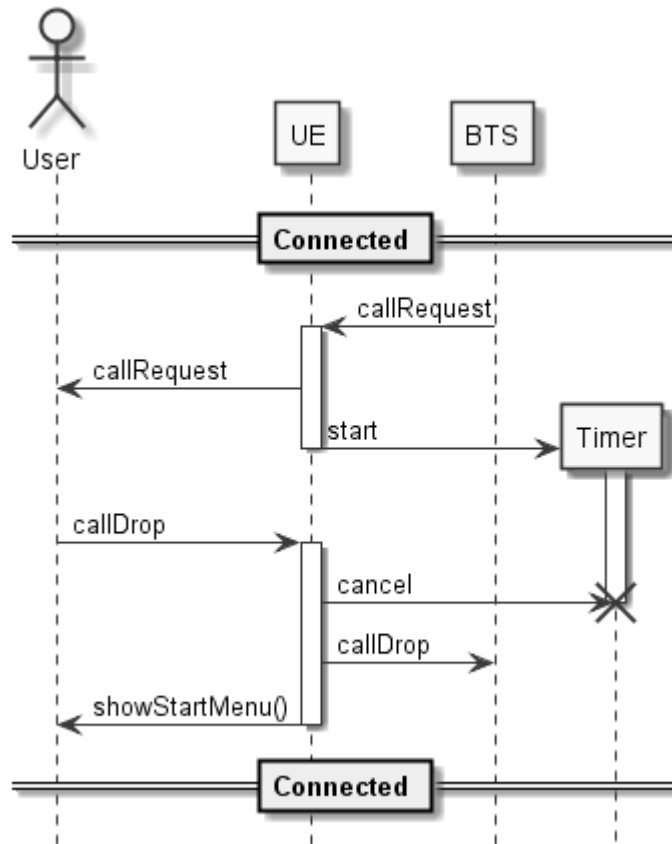


Figure 14 Received Call Request Rejected

4.1.6.4 Receiving Call Request Sequence Diagram – Failure (Timeout)

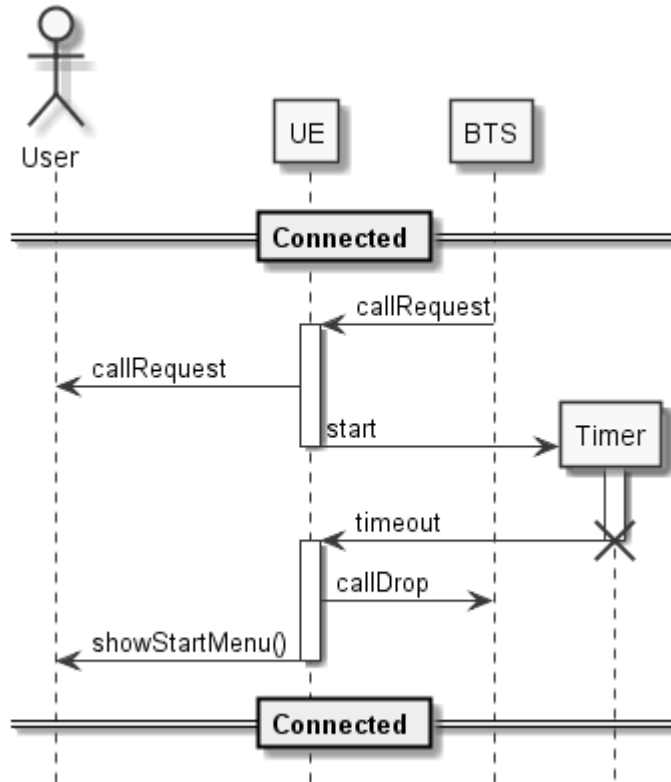


Figure 15 Received Call Request Expired

4.1.6.5 Receiving Call Request Sequence Diagram – Failure to Accept: Peer UE Disconnected

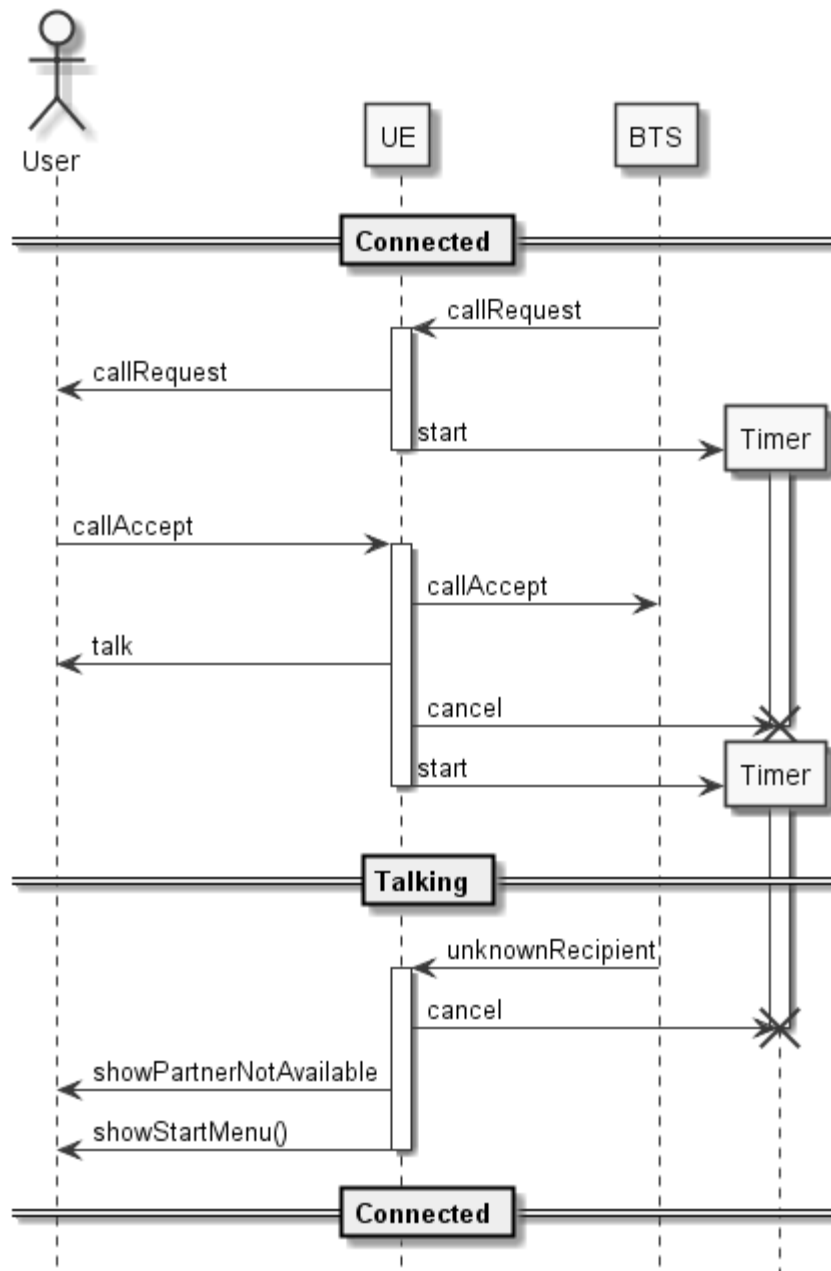


Figure 16 Received Call Request Accepted, but Peer UE already disconnected

4.1.7 Sending Call Request

UE shall allow User to initiate call to other User, via BTS, then other User UE.

While waiting for other User accepting the call, User shall be able to reject own request.

After successfully connected both User can talk.

This is mirror procedure to [Receiving Call Request](#).

Similarly to [Receiving Call Request](#), on every message sent to Peer UE - [Unknown Recipient](#) message can be received in response.

4.1.7.1 *Sending Call Request – Use Case*

Preconditions:

- UE in **Connected** state

Main success scenario:

1. User selects on UE: dial action (might be implement as menu/selection item)
2. UE switches to dialling mode
3. User types other UE Phone Number
4. User clicks Accept (green “pick up handset” button)
5. UE sends to BTS **CallRequest**(from=myPhoneNumber,to=enteredPhoneNumber)
6. BTS responds with **CallAccepted**(from=enteredPhoneNumber,to= myPhoneNumber)
7. UE changes its mode to conversation mode (**Talking** state)

Main failure scenario

6a. BTS responds with **CallDropped**

7a. UE informs User call dropped

7b UE returns to main menu mode

Alternative failure scenario – Peer User not connected

6a. BTS responds with **UnknownRecipient**

7a. UE informs User that peer UE is not connected

7b UE returns to main menu mode

Alternate failure scenario - Timeout

6a. BTS does not respond within 60s timeout

7a. UE informs User call dropped

7b UE returns to main menu mode

Alternate failure scenario - Resignation

6a. BTS is not responding, User click Reject (red “hang up the phone” button)

7a. UE sends to BTS **CallDropped**

7b. UE returns to main menu mode

Alternate failure scenario – Resignation and Peer UE was disconnected in meantime

6a. BTS is not responding, User click Reject (red “hang up the phone” button)

7a. UE sends to BTS **CallDropped**

7b. BTS responds with **UnknownRecipient**

7c. UE returns to main menu mode

Postconditions

- [success] UE is ready for talking (**Talking** state)
- [failure] UE returns to **Connected** state.

4.1.7.2 Sending Call Request Sequence Diagram – Success

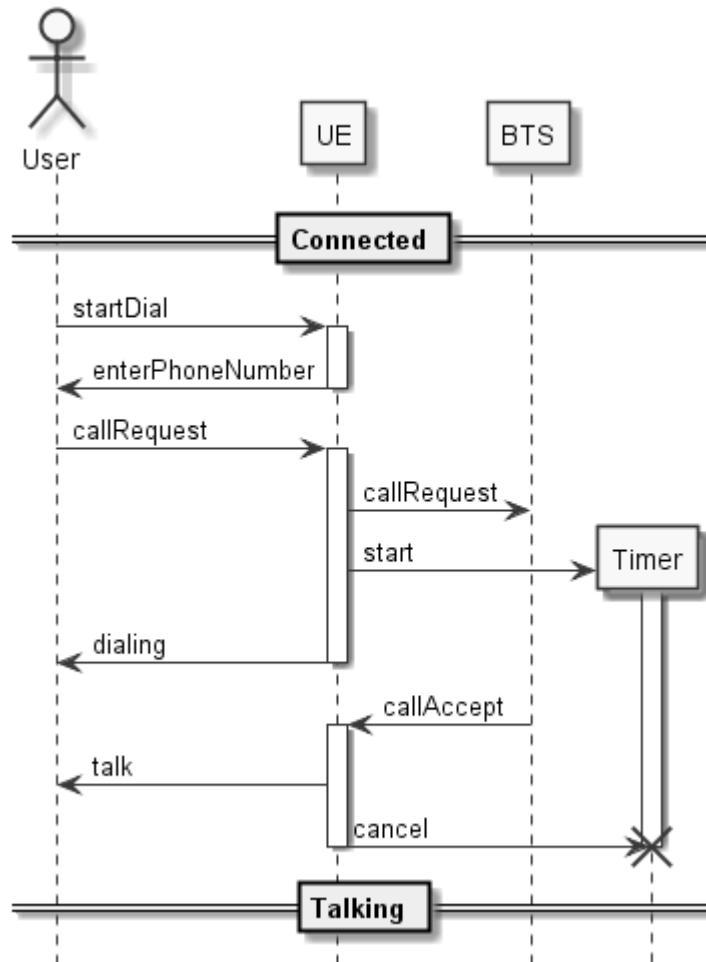


Figure 17 Call Initiated - Success

4.1.7.3 Sending Call Request Sequence Diagram – Failure: Peer User dropped call

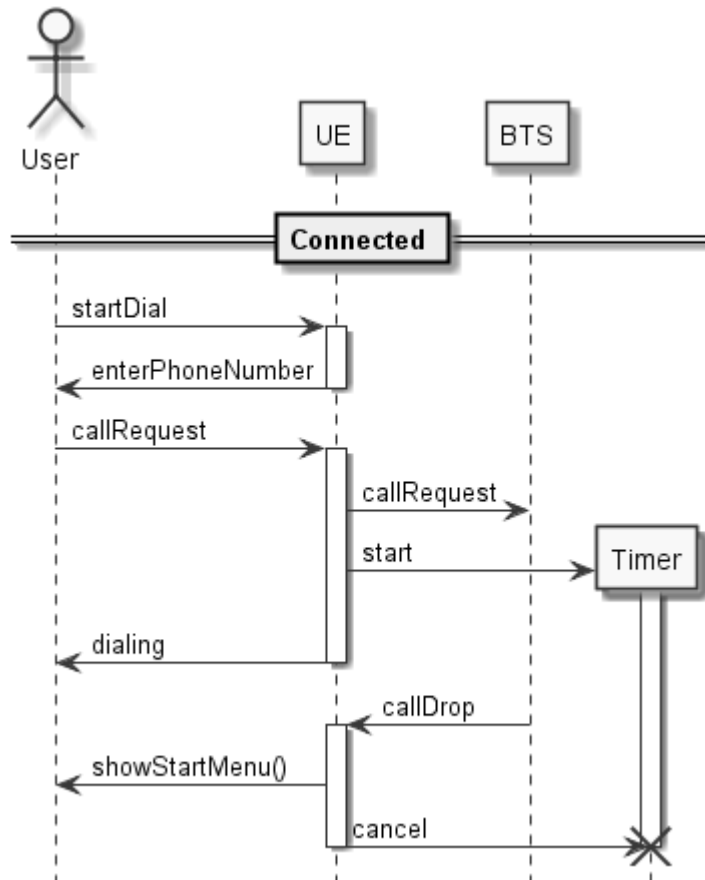


Figure 18 Call Initiated – Rejected

4.1.7.4 Sending Call Request Sequence Diagram – Failure: Peer User not connected

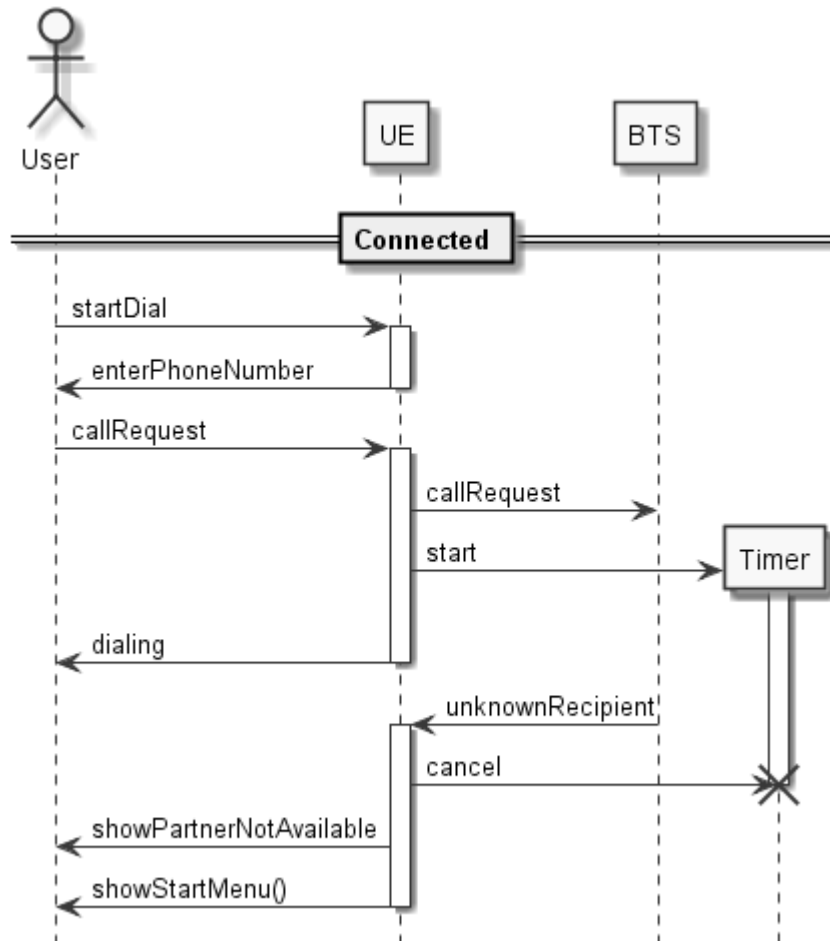


Figure 19 Call Initiated to not connected UE

4.1.7.5 Sending Call Request Sequence Diagram – Timeout

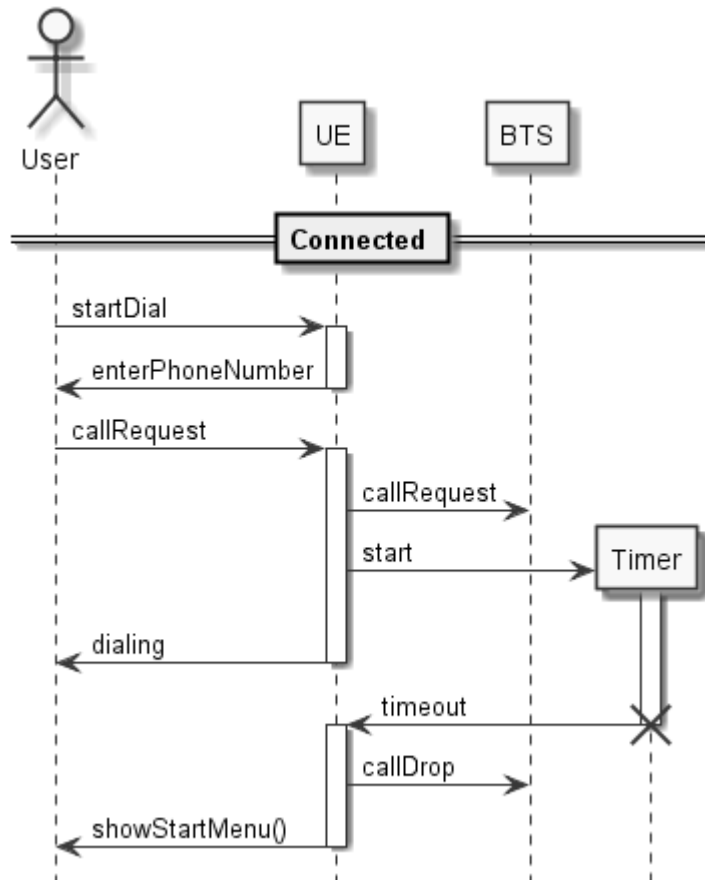


Figure 20 Call Initiated - Timeout

4.1.7.6 Sending Call Request Sequence Diagram – Resignation

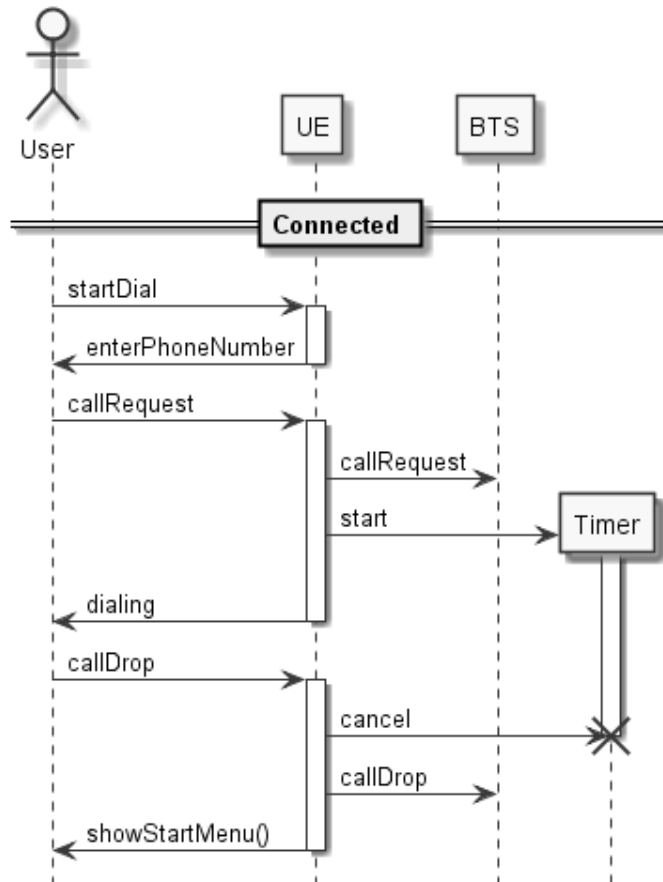


Figure 21 Call Initiated - Resignation

4.1.8 Dropping Call

User, when UE is in **Talking** state, shall be able to drop the call.

Similarly, when UE is in **Talking** state, on receiving **CallDropped** from other User shall end the current call.

Every **Unknown Recipient** message that is received on **CallDropped** message shall be ignored.

To keep this procedure simple – alternative scenarios where **UnknownRecipient** is received are not present in Sequence Diagrams.

4.1.8.1 Dropping Call – Use Case

Preconditions:

- UE in **Talking** state

Main scenario:

1. User clicks Reject (red “hang up the phone” button)
2. UE sends `CallDropped(from=myPhoneNumber, to=partnerPhoneNumber)` to BTS
 - 2.1 [Optional] `UnknownRecipient` is ignored.
3. UE shows main menu to User

Alternative scenario:

- 1a. BTS sends `CallDropped(from= partnerPhoneNumber, to= myPhoneNumber)` to UE
- 2a UE informs User that call ended
- 3 UE shows main menu to User

Postconditions

- UE is in `Connected` state.

4.1.8.2 Dropping Call Sequence Diagram – Initiated by User

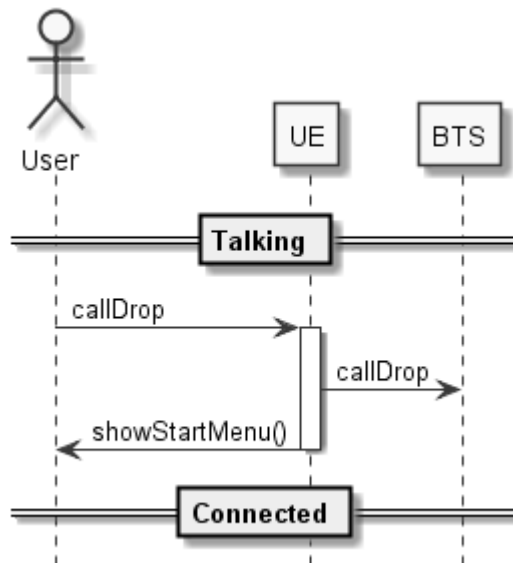


Figure 22 Call Dropped by User

4.1.8.3 Dropping Call Sequence Diagram – Initiated by other User

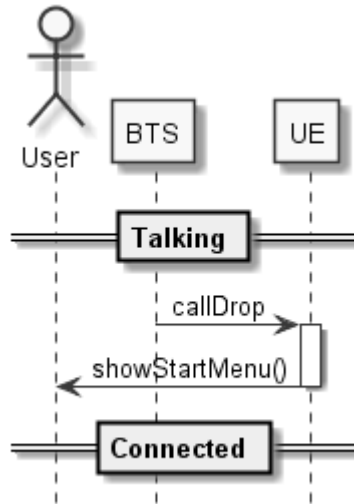


Figure 23 Call Dropped by other User

4.1.9 Call (talking)

UE is in **Talking** state.

User shall be able to enter message to the other User and UE shall send this message to other UE via BTS.

In parallel, on receiving **CallTalk** message from other User, UE shall presents the message to the User.

UE shall detect inactivity from both parties of the conversation.

4.1.9.1 Call (talking) – Use Case

Preconditions:

- UE in **Talking** state:

Main success scenario:

1. User enter text message
2. UE sends **CallTalk(partnerPhoneNumber, text)** to BTS
3. UE presents just sent message to User as part of conversation

Alternate success scenario

- 1a. no action
- 2a. UE receives **CallTalk(itsPhoneNumber, text)** from BTS

3a. UE presents just received message to User as part of conversation

Main failure scenario

4a. Last performed action was sending message to the other User

5a. No activity, as described in Success Scenarios, detected for 2min

6a. UE sends `CallDropped(from=myPhoneNumber, to=partnerPhoneNumber)` to BTS

7a. UE shows main menu to User

Alternate failure scenario

4b. Last performed action was receiving message from the other User

Alternate failure scenario – Peer UE was disconnected in meantime

2a. UE sends `CallTalk(partnerPhoneNumber, text)` to BTS

3a. BTS sends `UnknownRecipient(failedHeader.to=partnerPhoneNumber)` to UE

4a. UE shows to User, that Peer UE is disconnected

5a. UE shows main menu to User

Postconditions

- [success] UE is in `Talking` state.
- [failure] UE is in `Connected` state.

4.1.9.2 Call (talking) Sequence Diagram – Success – Sending talk message

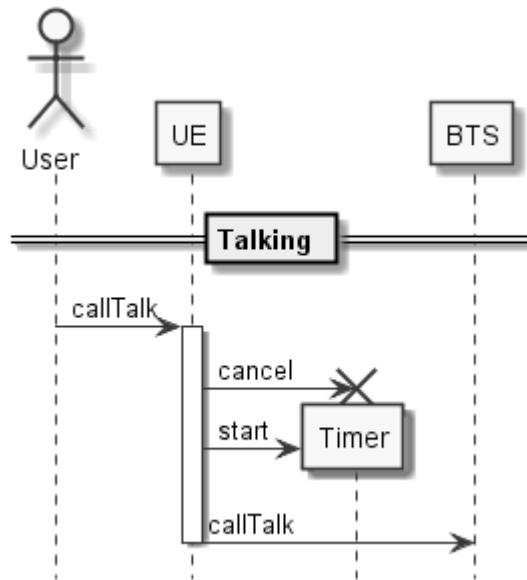


Figure 24 Call - Sending talk message

4.1.9.3 Call (talking) Sequence Diagram – Success – Receiving talk message

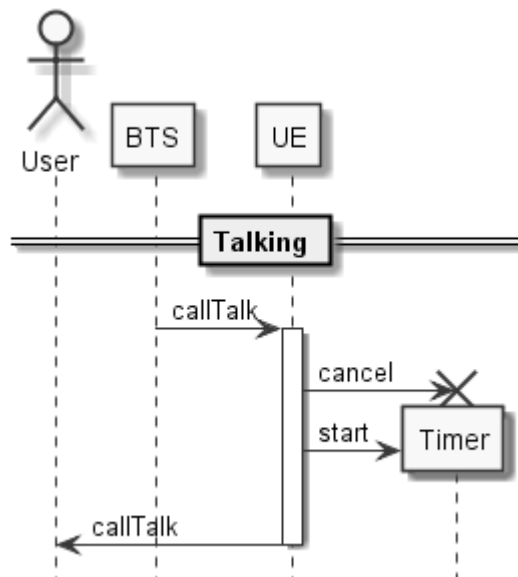


Figure 25 Call - Receiving talk message

4.1.9.4 Call (talking) Sequence Diagram – Failure – Timeout after sending message

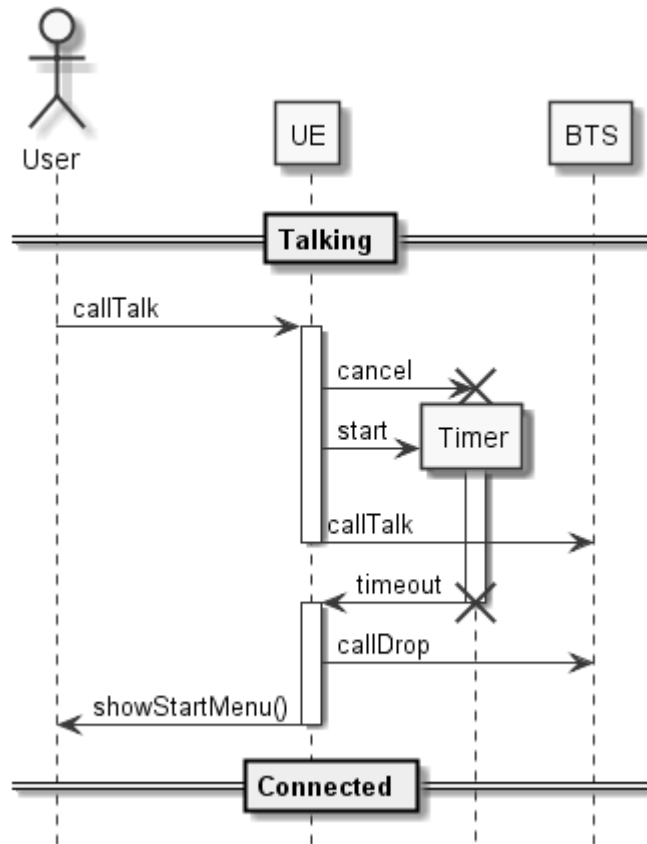


Figure 26 Call - Timeout (send)

4.1.9.5 Call (talking) Sequence Diagram – Failure – Timeout after receiving message

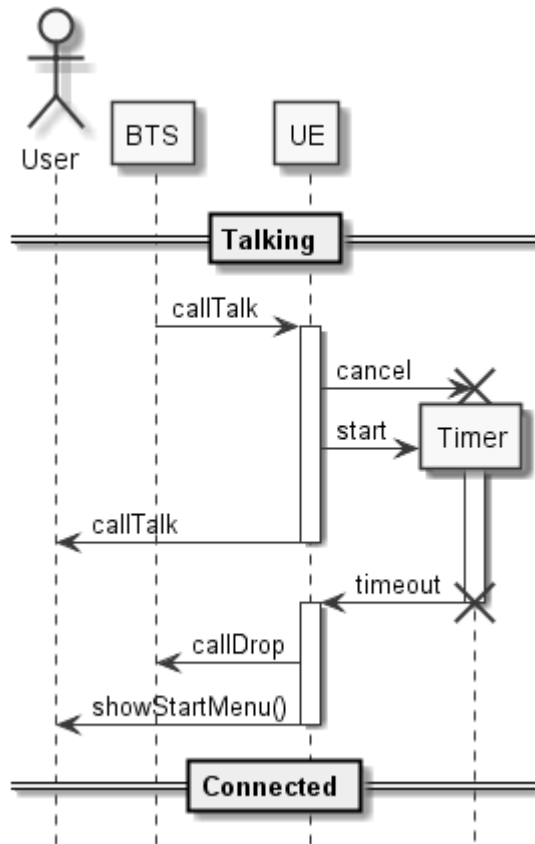


Figure 27 Call - Timeout (receive)

4.1.9.6 Call (talking) Sequence Diagram – Failure – Peer UE disconnected

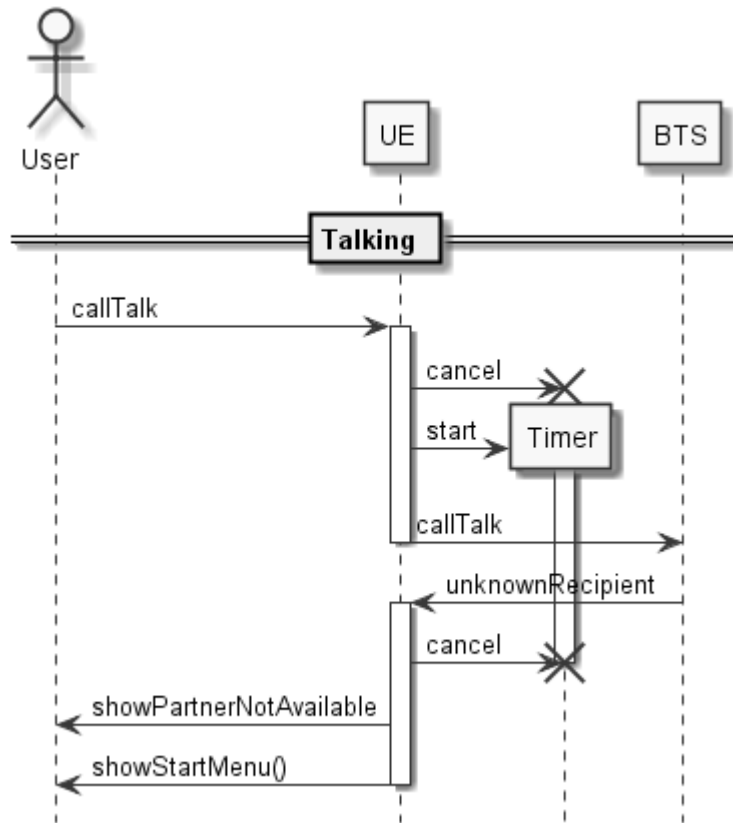


Figure 28 Call – Peer UE disconnected

4.2 UE basic interactions

Interactions between features defined in [UE basic features](#).

Only selected interactions are fully defined by using Use Cases or Sequence Diagrams.

Most of trivial interactions are only listed and very briefly described, but they shall be tested too.

It could be a good task for student to write missing UML diagrams...

4.2.1 Interaction with Attach to BTS procedure.

The only possible interactions with [Attach to BTS](#) procedure are in **Connecting** state, i.e. while UE is waiting on **AttachResponse**.

4.2.1.1 User closes UE while connecting to BTS – Use Case

Preconditions:

- UE in **Connecting** state:

Scenario:

1. User selects “close the application” action.
2. UE stops running timer
3. UE closes

Postconditions

- UE closed.

4.2.1.2 User closes UE while connecting to BTS - Sequence Diagram

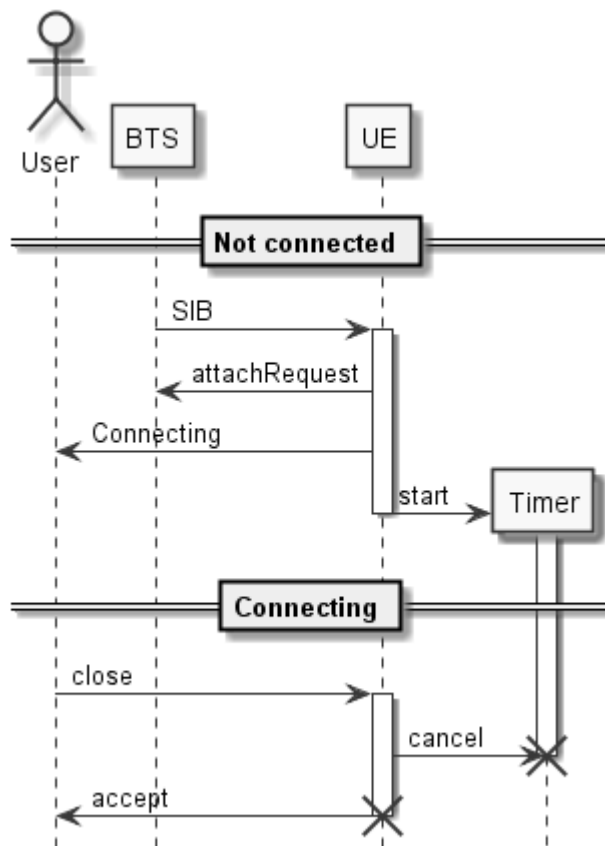


Figure 29 User closes UE while connecting to BTS

4.2.1.3 UE receives SIB while connecting to BTS – Use Case

Preconditions:

- UE in **Connecting** state

Scenario:

1. UE receives SIB from BTS
2. UE stores SIB data
 - 2.1 Note that current procedure is not restarted...

Postconditions

- UE in **Connecting** state

4.2.1.4 UE receives SIB while connecting to BTS Sequence Diagram

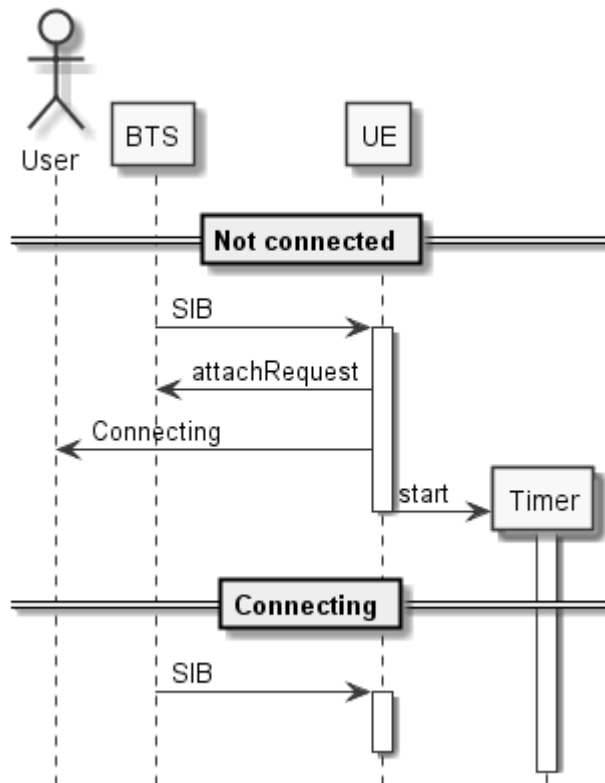


Figure 30 UE receives SIB while connecting to BTS

4.2.1.5 UE connection to BTS was dropped while connecting to BTS – Use Case

Preconditions:

- UE in **Connecting** state

Scenario:

1. UE ApplicationEnvironment informs (via callback) that connection to BTS is dropped
2. UE breaks Attach procedure (by cancelling Timer)

Postconditions

- UE in `NotConnected` state

4.2.1.6 UE connection to BTS was dropped while connecting to BTS – Sequence Diagram

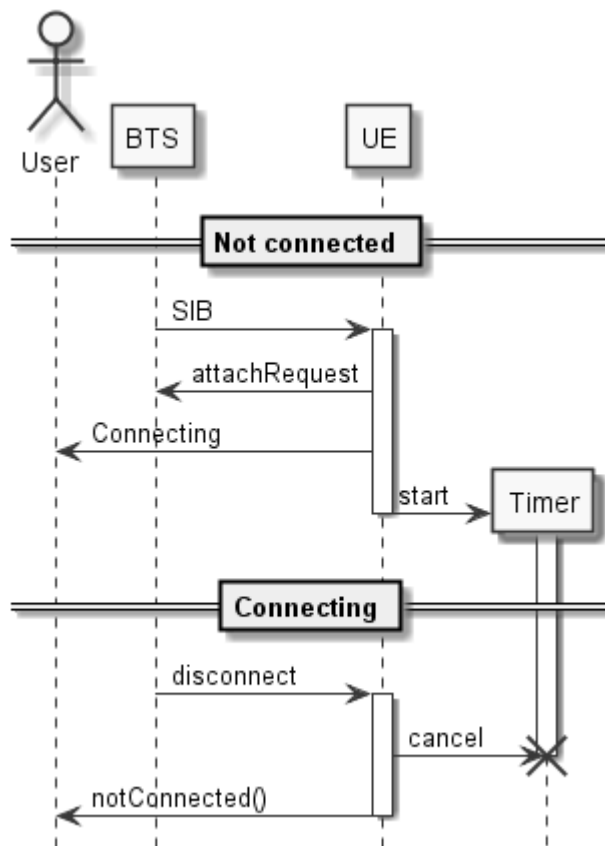


Figure 31 UE connection to BTS was dropped whilst attaching

4.2.2 Interactions with Re-Attach

As described in 4.1.2 Re-Attach to BTS – this is two steps procedure.

First part - Disconnecting – has no interactions as this is stateless procedure.

Second part – Attaching – has exactly the same interactions as initial attaching – see 4.2.1 Interaction with Attach to BTS procedure.

4.2.3 Interactions with Receiving SMS

There is no interactions with [Receiving SMS](#) scenario, because it is stateless procedure.

4.2.4 Interactions with Viewing SMS

There are two possible states of [Viewing SMS](#) procedure:

1. Viewing list of SMS
2. Viewing one selected SMS content.

The very same interactions scenarios apply for both states.

Just to clarify: interactions in both states should be verified by tests.

4.2.4.1 *User closes UE while viewing SMS/SMS list*

UE shall close immediately, i.e. it shall not wait, until User stops viewing SMS/SMS list.

4.2.4.2 *UE connection to BTS was dropped while viewing SMS/SMS List*

UE shall go to **NotConnected** state immediately, i.e. it shall not wait, until User stops viewing SMS/SMS list.

4.2.4.3 *UE receives SMS, while viewing SMS/SMS list*

UE shall store received SMS, but not interrupt the current viewing SMS scenario.

Basically, in the background of [Viewing SMS](#), the [Receiving SMS](#) procedure shall be performed.

It is **not** required, but can be implemented, that:

- The list of viewed SMS will be refreshed (by adding newly received SMS)
- The User will be informed that new SMS is received.

Of course, subsequent [Viewing SMS](#) procedure shall present all SMS, including the one discussed here.

4.2.4.4 *UE receives Call Request, while viewing SMS/SMS list – Use Case*

Preface:

[Receiving Call Request](#) procedure shall interrupt the [Viewing SMS](#).

Preconditions:

- UE in **ViewingSMS** state

Main Scenario:

1. BTS sends **CallRequest(from=callingPhoneNumber)** message to UE
2. UE stops ViewingSMS procedure
3. UE informs User about incoming call
4. Plus all steps as defined in [Receiving Call Request – Use Case](#)

5. There is no return to ViewingSMS, even in case of Call Request rejection

Alternate and Optional (not required to be implemented) Scenario:

5a. On [Receiving Call Request](#) failure scenarios – UE returns to [Viewing SMS](#) procedure,

Postconditions

- [main] UE not in [ViewingSMS](#) state – see [Receiving Call Request – Use Case](#)
- [alternate-optional] In case of Rejecting Call – UE in exactly the same [ViewingSMS](#) state

4.2.5 Interactions with Sending SMS

The interaction with [Sending SMS](#) are very similar to [Interactions with Viewing SMS](#).

4.2.5.1 User closes UE while sending SMS

UE shall close immediately, i.e. it shall not wait until User sends (or cancel sending) SMS.

4.2.5.2 UE connection to BTS was dropped while sending SMS

UE shall go to [NotConnected](#) state immediately, i.e. it shall not wait until sends (or cancel sending) SMS.

4.2.5.3 UE receives SMS, while sending SMS

UE shall store received SMS, but not interrupt the current sending SMS scenario.

Basically, in the background of [Sending SMS](#), the [Receiving SMS](#) procedure shall be performed.

It is **not** required, but can be implemented, that:

- The User will be informed that new SMS is received.

4.2.5.4 UE receives Call Request, while viewing SMS/SMS list – Use Case

Preface:

[Receiving Call Request](#) procedure shall interrupt the [Sending SMS](#).

Preconditions:

- UE in [SendingSMS](#) state

Main Scenario:

1. BTS sends [CallRequest\(from=callingPhoneNumber\)](#) message to UE
2. UE stops [SendingSMS](#) procedure
3. UE informs User about incoming call
4. Plus all steps as defined in [Receiving Call Request – Use Case](#)
5. There is no return to [SendingSMS](#), even in case of Call Request rejection

Alternate and Optional (not required to be implemented) Scenario:

5a. On [Receiving Call Request](#) failure scenarios – UE returns to [Sending SMS](#) procedure,

Postconditions

- [main] UE not in [SendingSMS](#) state – see [Receiving Call Request – Use Case](#)
- [alternate-optional] In case of Rejecting Call – UE in exactly the same [SendingSMS](#) state

4.2.6 Interactions with Receiving Call Request

This chapter describes interaction with [Receiving Call Request](#) procedure.

So, this is the state when UE is waiting on User to accept (or reject, or timeout) the incoming call – e.g. see [Figure 13 Received Call Request Accepted](#).

4.2.6.1 User closes UE while receiving CallRequest

UE shall close without waiting on completion of [Receiving Call Request](#) procedure.

UE shall send the corresponding [Call Dropped](#) message to BTS, as it is showed in [Figure 14 Received Call Request Rejected](#).

4.2.6.2 UE connection to BTS was dropped while receiving CallRequest

UE shall go to [NotConnected](#) state immediately.

In this very interactions there is no possibility to inform peer UE (via [Call Dropped](#)) that Call cannot be performed.

4.2.6.3 UE receives SMS, while receiving CallRequest

UE shall store received SMS, but not interrupt the current receiving Call Request scenario.

Basically, in the background of [Receiving Call Request](#), the [Receiving SMS](#) procedure shall be performed.

It is **not** required, but can be implemented, that:

- The User will be informed that new SMS is received.

4.2.6.4 UE receives subsequent Call Request

UE shall drop the new incoming Call Request.

UE shall respond on new Call Request with [Call Dropped](#), unless it is Call Request from the same UE as the old Call Request.

4.2.7 Interactions with Sending Call Request

This chapter describes interaction with [Sending Call Request](#).

This procedure consists of two main steps:

1. User is typing phone number – so before User clicks Accept (green “pick up handset” button)

2. UE is waiting on **Call Accepted** message from BTS (actually, from other UE/User)

When this is needed – UE handles interruptions in these steps differently.

4.2.7.1 User closes UE while sending Call Request

UE shall close without waiting on completion of [Sending Call Request](#) procedure.

If User already clicks Accept (green “pick up handset” button) and **Call Request** message was sent, the corresponding **Call Dropped** should be sent to BTS, as it is showed in **Figure 21 Call Initiated - Resignation**.

4.2.7.2 UE connection to BTS was dropped while sending Call Request

UE shall go to **NotConnected** state immediately.

In this very interactions there is no possibility to inform peer UE (via **Call Dropped**) that Call cannot be performed.

4.2.7.3 UE receives SMS, while sending Call Request

UE shall store received SMS, but not interrupt the current sending Call Request scenario.

Basically, in the background of [Sending Call Request](#), the [Receiving SMS](#) procedure shall be performed.

It is **not** required, but can be implemented, that:

- The User will be informed that new SMS is received.

4.2.7.4 UE receives Call Request, while sending Call Request – Use Case

Preface:

[Receiving Call Request](#) procedure shall interrupt the [Sending Call Request](#).

Preconditions:

- UE in **SendingCall** state

1st Scenario – CallRequest was not sent to BTS:

1. BTS sends **CallRequest(from=callingPhoneNumber)** message to UE
2. UE stops **SendingCallRequest** procedure
3. UE informs User about incoming call
4. Plus all steps as defined in [Receiving Call Request – Use Case](#)
5. There is no return to **SendingCallRequest**, even in case of Call Request rejection

2nd Scenario – CallRequest was sent to BTS:

1. BTS sends **CallRequest(from=callingPhoneNumber)** message to UE
2. UT stops **SendingCallRequest** procedure
 - 2.1 UE sends **CallDropped(from=itsPhoneNumber)** to BTS

3. UE informs User about incoming call
4. Plus all steps as defined in [Receiving Call Request – Use Case](#)
5. There is no return to SendingCallRequest, even in case of Call Request rejection

Postconditions

- [main] UE not in [SendingCall](#) state – see [Receiving Call Request – Use Case](#)

4.2.8 Interactions with Dropping Call

There is no interactions with [Dropping Call](#) scenario, because it is stateless procedure.

4.2.9 Interactions with Call (Talking)

This chapter describes interaction with [Call \(talking\)](#).

4.2.9.1 User closes UE while having Call (Talking)

UE shall close without waiting on completion of [Call \(talking\)](#) procedure.

The corresponding [Call Dropped](#) message should be sent to BTS, as it is showed in [Figure 22 Call Dropped by User](#).

4.2.9.2 UE connection to BTS was dropped while having Call (Talking)

UE shall go to [NotConnected](#) state immediately.

In this very interactions there is no possibility to inform peer UE (via [Call Dropped](#)) that Call cannot be continued.

4.2.9.3 UE receives SMS, while having Call (Talking)

UE shall store received SMS, but not interrupt the current sending Call(Talking) scenario.

Basically, in the background of [Call \(talking\)](#), the [Receiving SMS](#) procedure shall be performed.

It is **not** required, but can be implemented, that:

- The User will be informed that new SMS is received.

4.2.9.4 UE receives Call Request, while having Call (Talking)

UE shall drop the new incoming Call Request.

UE shall respond on new Call Request with [Call Dropped](#), unless it is Call Request from the same UE as the ongoing Call (Talking).

4.3 UE advanced features

Advanced features are discussed in details during lectures.

Examples of advanced features:

1. Sending encrypted SMS (see [Table 8 SMSSMS](#) message)
2. Request Call (Talking) in encrypted mode (see [Call Request](#) message)

5 List of figures

Figure 1 Project directory structure.....	6
Figure 2 Possible deployment	7
Figure 3 UE directory structure.....	8
Figure 4 UE Application States.....	11
Figure 5 Attach to BTS - Success.....	19
Figure 6 Attach to BTS, BTS rejected.....	20
Figure 7 Attach to BTS, timeout	21
Figure 8 Receiving SMS.....	23
Figure 9 Viewing SMS.....	24
Figure 10 Sending SMS – Success.....	25
Figure 11 Sending SMS – Partial Success	26
Figure 12 Sending SMS - Abort.....	27
Figure 13 Received Call Request Accepted	29
Figure 14 Received Call Request Rejected	30
Figure 15 Received Call Request Expired	31
Figure 16 Received Call Request Accepted, but Peer UE already disconnected.....	32
Figure 17 Call Initiated - Success.....	35
Figure 18 Call Initiated – Rejected.....	36
Figure 19 Call Initiated to not connected UE	37
Figure 20 Call Initiated - Timeout	38
Figure 21 Call Initiated - Resignation.....	39
Figure 22 Call Dropped by User	40
Figure 23 Call Dropped by other User	41
Figure 24 Call - Sending talk message.....	43
Figure 25 Call - Receiving talk message	43
Figure 26 Call - Timeout (send).....	44
Figure 27 Call - Timeout (receive)	45
Figure 28 Call – Peer UE disconnected.....	46
Figure 29 User closes UE while connecting to BTS.....	47
Figure 30 UE receives SIB while connecting to BTS	48
Figure 31 UE connection to BTS was dropped whilst attaching.....	49

6 List of tables

Table 1 SIB (System Information Broadcast) message	12
Table 2 Attach Request.....	13
Table 3 Attach Response	13
Table 4 Message header from/to fields meaning when message is sent from UE	13
Table 6.1 Encryption for XOR encrypted communication	14
Table 6.2 Encryption for Cesar encrypted communication.....	15
Table 6.3 Encryption for RSA encrypted communication	15
Table 7 SMS	15
Table 8 Call Request.....	16
Table 9 Call Accepted	16
Table 10 Call Dropped	16
Table 11 Call Talk	16
Table 12 Unknown Recipient.....	17
Table 13 Unknown Sender.....	17

7 List of acronyms

LTE	Long Term Evolution (3GPP 4G technology)
3GPP	3rd Generation Partnership Project
BTS	Base Transceiver Station
UE	User Equipment
NE	Network Element
SMS	Short Message Service
TCP	Transmission Control Protocol
IP	Internet Protocol