Signalling method for mobile communications network

Badoui Y. Heik, Rahim Tafazolli

Center for Communication Systems Research, University of Surrey, Surrey GU2 7XH, UK
E-mail: B.Yamine-Heik@surrey.ac.uk

Published in *The Journal of Engineering*; Received on 14th October 2014; Accepted on 24th November 2014

Abstract: In this study, a novel and generic signalling method is proposed that results in a significant reduction in signalling messages exchanged between a user equipment and the network over the air interface. Reduction of signalling messages in future mobile cellular systems is an important issue because of the current trend of deploying smaller cells while achieving higher than expected data area capacities. To demonstrate the effectiveness of this method within the standard constraints, some signalling procedures of the 3rd generation mobile standard, Universal Mobile Telecommunication System (UMTS), are investigated. In one application, the number of UMTS packet switch call setup messages was reduced down to three messages only. In such a scenario, measurements show that the gains in comparison with prior call setup is 61% in the duration of the call setup, 50% in the downlink processing delays, 93% in the downlink radio power transmission and 79% in the uplink radio power transmission.

1 Introduction

In any system, the higher size level of signalling messages of any procedure means less bandwidth left for traffic, more latency for procedure setup and more processing on all of the equipment involved in that procedure. Keeping in mind that the size of the signalling messages matters, but the number of signalling messages involved is a big contributor to additional latency and processing in the network. In this paper, a novel generic storage method called the dynamic storage method (DyStoM) is proposed. It reduces the size and also the number of messages in any signalling procedure by looking at the repetitive information in a signalling procedure in order to store them during the first run and avoid sending them again during a following procedure run. An example of static information is the data channel configuration of the user, whereas an example of dynamic information is the radio channel resources.

On the basis of the presence of repetitive information in the wireless system, different methods have already been proposed in prior articles. In [1], two different methods are described. The first one is based on hardcoding the static information at user equipment (UE) and at radio network controller (RNC). The second one consists of broadcasting information, on the air interface, that will be used later by the UE during any handover, e.g. from GSM, to universal mobile telecommunications system (UMTS). In [2], a method consists of comparing a configuration to send to the UE with a hardcoded configuration at the RNC and at the UE. If the difference is small only that difference is sent with the identity of the stored one, otherwise the complete configuration is sent. In [3], broadcast information is used to send information in advance, not handover messages like in [1], but instead a part of a signalling message that will be exchanged later during any upcoming UE call setup. In [4], repetitive information consisting of parts of signalling messages of the UMTS call setup is stored on an external server. When any change occurs to the stored information, the UE is notified via the air interface and the UE triggers a connection to that server to obtain the latest update. In [5], another method is used to store the repetitive information dynamically at UE and at the RNC, but this method uses the broadcast channel on the air interface as a means to achieve that storage. As yet another option in prior articles shows in [6] a new method consists of setting a timer on the receiver and/or transmitter. As long as this timer does not run out, both equipment exchanges receive a reduced radio resource control (RRC) message, otherwise a full message is sent. In [7] and for some particular type of calls like push-to-talk over cellular, the contents of the message that carries the data user traffic configuration in UMTS call setup, are embedded in the message that carries signalling channel configuration. In other words, the RRC radio bearer setup (RBS) is embedded in the RRC connection setup (CS). To reduce the size of the final message, the hardcoding of the contents of both messages is proposed. However, the major drawback of that method is that any call with that method will go non-encrypted over the air interface.

It should be noted that the hardcoding solution is not practical, because on one hand it forces all vendors and operators to use the same values for all stored parameters, and on the other hand any change in the stored configuration would mean a software upgrade. That is why all the other methods described in [1–7] were proposed. However, there are two main disadvantages with the solutions in the prior articles. First, most if not all of them use the air interface as an input for their algorithm in order to update the stored configuration. This means that these methods do not apply between two pieces of equipment where there is no air interface. Second, all of these methods only store a part of one signalling message, whereas with the proposed method not only one part of the message could be stored but more importantly parts of different messages could be embebbed into one message. In addition, with the proposed method parts of different messages could be embedded into one message. To the knowledge of the authors there is no such approach in prior articles.

2 Dynamic storage method

Fig. 1 is an example of a signalling procedure exchanged between two pieces of equipment.

The objective of DyStoM is to store part of the information of that procedure during the first run in order to exchange only the remaining information during the second procedure run. A flowchart of that method is shown in Fig. 2, and then a description of each step is described in the following paragraphs:

Step 1: Repetitive procedure, information and actions: In any system, in order to apply the DyStoM one should look at the occurrence of any type of repetitive procedure, information or action. An example of a repetitive procedure would be a call setup. An example of repetitive information is the contents of exchanged messages, and an example of a repetitive action would be a user moving in the same geographical area everyday.

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)

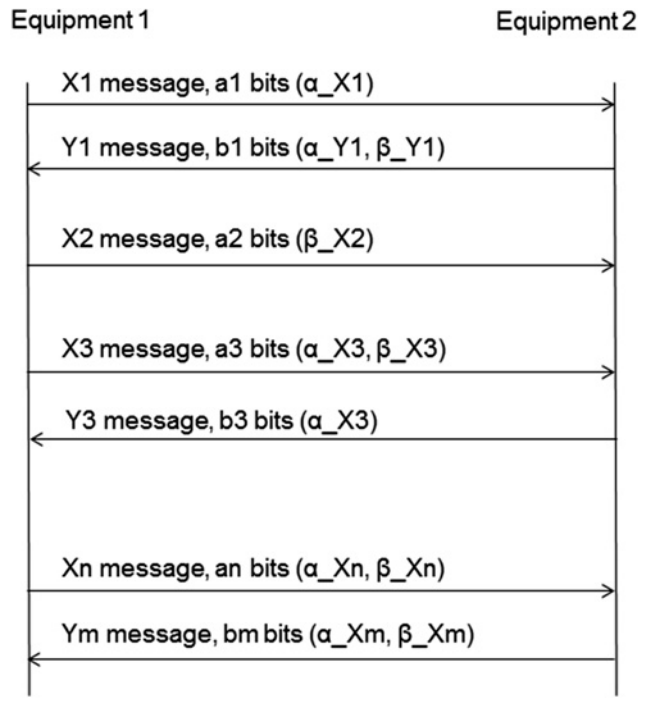

Equipment 1
Equipment 2
X1 message, a1 bits (α_X1)
Y1 message, b1 bits (α_Y1, β_Y1)
X2 message, a2 bits (β_X2)
X3 message, a3 bits (α_X3, β_X3)
Y3 message, b3 bits (α_X3)
Xn message, an bits (α_Xn, β_Xn)
Ym message, bm bits (α_Xm, β_Xm)

Fig. 1 *Messages contents during first call*

Step 2: Divide each message into two parts α and β: Each message of the procedure is divided into two parts called α and β. Part α contains the part of the message where the values of some, if not all, parameters change every time the procedure is executed. Part β contains the remaining part of the message, where the values of the parameters remain fixed every time the procedure is executed. As a result β part could be stored whereas part α should not.

Step 2.1: Define a server and client(s): A server is the only piece of equipment that has the authority to change the value of β at any time. Contrary to this, a client node is the piece of equipment where the value of β cannot be changed. It is up to the server to update the client with the new value of β. Depending on the system that is used, the equipment that is involved in the signalling procedure could be acting as a server or as a client.

In a first scenario, one piece of equipment acts as a server and the other piece of equipment acts as a client. This scenario is called ‘server to client’. In a second scenario, each piece of equipment acts as a server. This scenario is called ‘server to server’.

Steps 3: Storing β on server and client: For any signalling procedure exchanged between two or more pieces of equipment, DyStoM is mainly used in the following three scenarios:
1. Store β of one message during the first call and only send its α part during following calls. An application to UMTS is described in [8].
2. Store a complete message during the first call and skip sending it during following calls. An application to UMTS is described in [9].
3. Store βi of different messages during the first call, and then send all their αi in a much lower number of messages during the second call, ideally in one message. An application to UMTS is studied in this paper.

Step 4: Evaluation: It is important to note that for the sake of efficiency β part is stored when it makes a reasonably large part of the total signalling procedure.
The number of signalling messages is reduced whenever there is any benefit in terms of latency and/or interference, processing, transmission power etc.

Step 5: ‘Open interface’ or ‘closed interface’ based on client address change: The ‘open interface’ is a system where the address of the server remains fixed over time, whereas the address of the identity of all peer nodes, called clients, changes, whereas in a ‘closed interface’ all pieces of equipment involved in the signalling procedure, server and client(s), have a fixed address that does not change in time. An example of the application of ‘closed interface’ to UMTS is described in [10], where steps 7.1, 7.2 and 7.3 shown in Fig. 2 are described. In this paper, only the case of ‘open interface’ is studied.
Step 6: ‘Open interface’ method: Below all of the steps of method 1 are described, and these are step 6.1, 6.2, 6.3 and 6.4 as shown in Fig. 2.

Step 6.1: Define tag on server and clients: A new parameter called a tag is defined on piece of each equipment. That tag is an integer coded in a certain number of bits, for example, 8 bits. Each value of tag on a piece of equipment represents a certain configuration of βi on that piece of equipment. If both pieces of equipment have the same value of tag, then that means they both have the same configuration of βi, meaning the same values for all parameters of βi. Otherwise, each piece of equipment has a different configuration of βi. The tag on the server is called the serverk_*βi*_tag, where each k represents one server, and on a client it is called the clientj_*βi*_tag, where each j represents one client.
Step 6.2: Define rules on tag: Suppose that at t1, if the client and server each have a tag equal to 5, the client is switched off or is moved temporarily to another network or goes out of service for a long period T. Also suppose that during that period T the operator changes the value of βi many times in a way that means that at a certain time during T the value of the tag on the server is again 5. In a case where the tag is coded in 8 bits this would occur at time t2, after 256 changes of βi. Now suppose that the client that went away from the network with a value tag equal to 5 comes back to the network at time t2. The value of the tag on client is equal to 5 and is the same as on the server; however, the βi on the server is different than the βi at the client. To avoid such a situation of confusion, different rules on tags are defined, as shown below.

Rule 1: Initially, the tag on all client(s) is equal to zero, whereas the tag on the server should not take, at any time, the value of 0, but starts with the value of 1. In that way, during the first call and according to the algorithm in step 6.4 below, βi is always sent to the client by the server as the tag on the client is equal to 0 and is different than the tag on the server which is equal to 1.
Rule 2: Whenever the client is switched off/on or moved to another network, the tag on the client is set to 0. In this way, when the client comes back to the network, the most updated βi will be sent by the server, which has a tag value that is always >0 by definition of tag Rule 1.
Rule 3: The value of the serverj_*βi*_tag is incremented by 1 each time the value of one parameter of any βi is changed on the server side.
Rule 4: A new timer called the serverj_timer is defined on the server j side. During that timer the value of the serverj_*βi*_tag [1, 255] could not take the same value twice.
Rule 5: A new timer called the clientk_timer is defined at client k. When that timer expires the clientk_*βi*_tag is set to 0.

Note that a synchronisation between the timer on the server and the timer on the client is necessary in order to avoid any confustion with the tag values at the client and at the server. This might happen for example if the client is a UE that comes back to the network after losing synchronisation for a while e.g. the UE goes out of coverage or the UE goes roaming.

In this paper the timers synchronisation is done as following:

First, the value of serverj_timer is sent to the client e.g. via a dedicated message or via other ways like over the air interface. Later the first call, the value of clientk_timer is defined as equal to [(severj_timer)–tcall]. Then during the second and the following call clientk_timer = severalj_timer in other words tcall is just a temporary value that is used to synchronize clientk_timer & serverj_timer. Tcall is defined as following: when the client sends a request message, denoted 'UL request', to the server, the value of tcall is sent to the client via a dedicated message, denoted 'DL response', where tcall is calculated as follows: It is equal to the timestamp of the 'DL response', minus the timestamp of the most recent start time of timerj_server at the server. For example, if serverj_timer starts at 12:00 for a duration of 1 hour and if 'DL response' is sent at 12:40 then tcall is simply equal to 40 minutes and clientk is equal to 20 minutes during the first call. Later during the second & following calls clientk_timer becomes equal to the serverj_timer = 1 hour.

Step 6.3: Define 'request procedure' and 'configuration procedure'. The DyStoM algorithm simply consists of comparing the value of the tag on the client with the value of the tag on the server. If they are equal, the server only sends the α part, otherwise it sends the messages as demonstrated in prior articles.

Two cases are studied.

In the first case, in order for that algorithm to work the server needs to know the value of the tag at the client side. This is done by the client sending the tag value to the server, for example, in a message, before the algorithm makes a decision. This scenario is called a 'request procedure'.

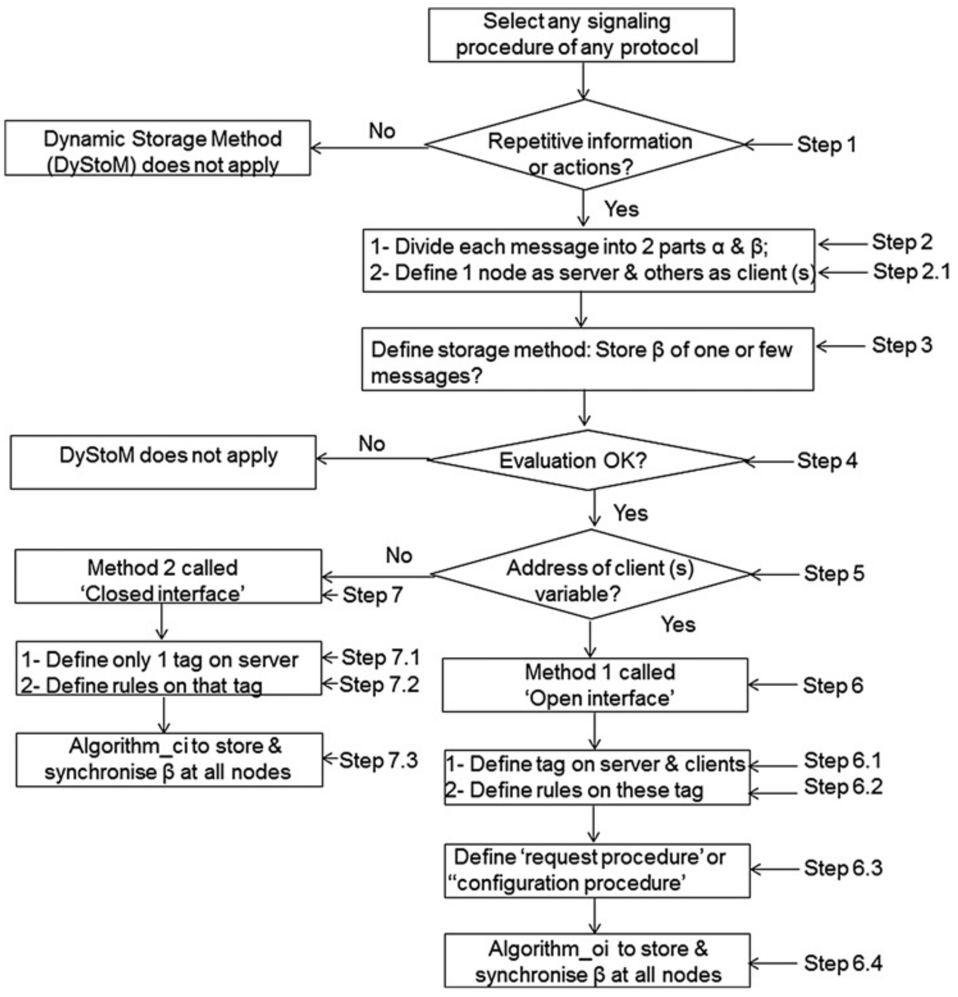
On the other hand, in a second case it is possible that many signalling procedures can be composed of only two messages or perhaps more where the message to be reduced is the first message that is sent from the server to the client. In this paper, this procedure is called a 'configuration procedure'. In this scenario, the server does not know the value of the tag at the client side, and therefore it cannot compare the tag value of the client with the tag value at the server. As a result, the proposed algorithm in this paper does not apply in the case of a 'configuration procedure'. A new algorithm is required for that purpose and is a subject for future study.

Furthermore, in some procedures the server can transfer information to the client, but the client does not acknowledge or communicate the version of the stored configuration to the server. An example of such a procedure is when the UE is in idle mode and listening to the information being broadcast from the radio base station. Again, this procedure is similar to the case of the configuration procedure and its storage algorithm is left for future study.

Step 6.4: Algorithm_oi to store and synchronise β at all nodes. Another parameter called the full_size_message is introduced, and this is coded in 1 bit. If the value is 0, it means that the signalling procedure is sent in the manner described in the previous articles, otherwise the reduced version of the signalling procedure is sent. The flowchart is shown in Fig. 3 below. Note that this parameter could be sent in either the message which needs to be reduced or in any message before.

3 Reducing the number of RRC messages in UMTS call setup down to three

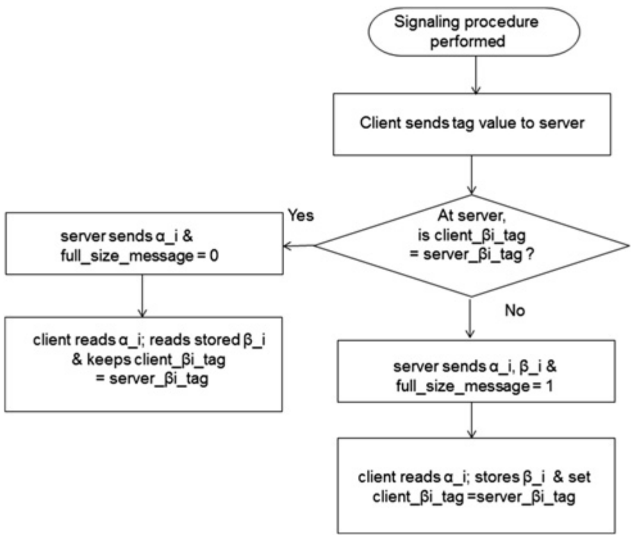
In Fig. 4, the UMTS packet switch call described in prior articles is shown. Note that for a speech call the call flow is slightly different; however, the same method and calculations apply.



Select any signaling procedure of any protocol
Repetitive information or actions?
No
Dynamic Storage Method (DyStoM) does not apply
Step 1
Yes
1- Divide each message into 2 parts α & β;
2- Define 1 node as server & others as client (s)
Step 2
Step 2.1
Define storage method: Store β of one or few messages?
Step 3
Evaluation OK?
No
DyStoM does not apply
Step 4
Yes
Address of client (s) variable?
No
Step 5
Method 2 called 'Closed interface'
Step 7
1- Define only 1 tag on server
2- Define rules on that tag
Step 7.1
Step 7.2
Algorithm_ci to store & synchronise β at all nodes
Step 7.3
Yes
Method 1 called 'Open interface'
Step 6
1- Define tag on server & clients
2- Define rules on these tag
Step 6.1
Step 6.2
Define 'request procedure' or "configuration procedure'
Step 6.3
Algorithm_oi to store & synchronise β at all nodes
Step 6.4

Fig. 2 *Flowchart of DyStoM*

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)



Signaling procedure performed
Client sends tag value to server
At server, is client_βi_tag = server_βi_tag ?
Yes
No
server sends α_i & full_size_message = 0
client reads α_i; reads stored β_i & keeps client_βi_tag = server_βi_tag
server sends α_i, β_i & full_size_message = 1
client reads α_i; stores β_i & set client_βi_tag = server_βi_tag

Fig. 3 *'Open interface' algorithm flowchart*

3.1 Steps to reduce UMTS call setup down to three messages

This section is composed of four steps: A1, A2, A3 and A4 as shown in Fig. 5.

Step A1: In UL and DL, divide each message i into two parts α_i and β_i. Each RRC message of the call setup i is divided into two parts α_i and β_i, as defined in step 2 in Section 2, then the contents of each part are called information elements (IE) and are shown in tables below.

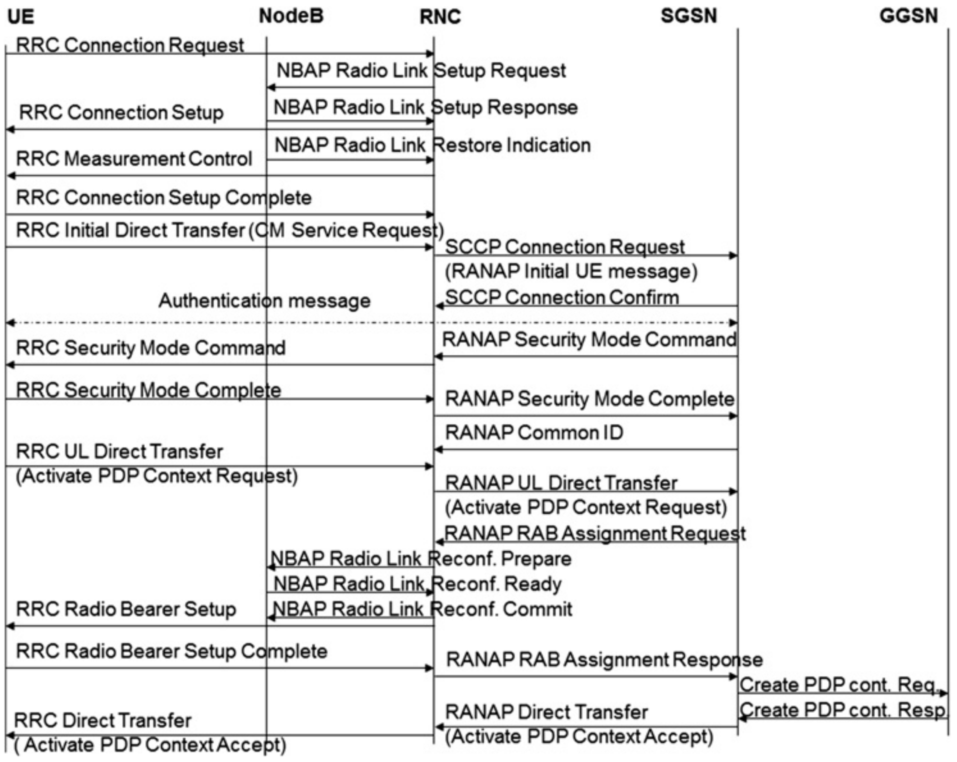
The way that the IEs are placed in α_i or β_i in the tables represents a working configuration. However, these IE's could be further optimised by each vendor, for example, some IE in the α part could be moved to the β part and vice versa. For example, one non-access stratum (NAS) message like an activate PDP context request (APDPR) or activate PDP context accepted (APDPA) is carried via one RRC message, called the direct transfer, between the UE and RNC, and then via one radio access network application part (RANAP) [11] message between the RNC and core network. The size of the RRC message carrying the NAS is very small. That is why the size of such RRC messages is not considered in this paper. Instead the α_i and β_i of the NAS messages are shown in the tables below and studied.

Step A2: Group all DL α_i in one message: New_APDPA. In prior articles there are five messages which are sent in the DL direction, as shown in Fig. 4. For two of them, the 'CS' and the 'RBS' messages, the tables of the α and β parts are shown in [8]. For the 'security mode command' (SMC) and the APDPA messages, α and β are shown in Tables 1 and 2 below. On the other hand, we consider in this paper that some messages are composed only of the β part. The contents of these messages are not shown in the tables below because they are stored during the first call and their size is not considered in the calculations below. The 'measurement control' is one example of such messages.

On the basis of the study and the observation of the timestamps of an UMTS call setup and in order to obtain the maximum gain of DyStoM, the α parts of all DL messages are sent in one message called the new_APDPA, which is demonstrated as follows

$$\text{new\_APDPA} = \alpha\text{\_CS} + \alpha\text{\_RBS} + \alpha\text{\_SMC} + \alpha\text{\_APDPA} \quad (1)$$

In practice, this requires the transmitter machine to delay the transmission of different signalling messages and then combine them in one message. The same principle is followed for the UL messages described in the following step A3.



UE
NodeB
RNC
SGSN
GGSN
RRC Connection Request
NBAP Radio Link Setup Request
RRC Connection Setup
NBAP Radio Link Setup Response
NBAP Radio Link Restore Indication
RRC Measurement Control
RRC Connection Setup Complete
RRC Initial Direct Transfer (CM Service Request)
SCCP Connection Request
(RANAP Initial UE message)
Authentication message
SCCP Connection Confirm
RANAP Security Mode Command
RRC Security Mode Command
RRC Security Mode Complete
RANAP Security Mode Complete
RANAP Common ID
RRC UL Direct Transfer
(Activate PDP Context Request)
RANAP UL Direct Transfer
(Activate PDP Context Request)
RANAP RAB Assignment Request
NBAP Radio Link Reconf. Prepare
NBAP Radio Link Reconf. Ready
RRC Radio Bearer Setup
NBAP Radio Link Reconf. Commit
RRC Radio Bearer Setup Complete
RANAP RAB Assignment Response
Create PDP cont. Req.
Create PDP cont. Resp
RRC Direct Transfer
(Activate PDP Context Accept)
RANAP Direct Transfer
(Activate PDP Context Accept)

Fig. 4 *UMTS call setup in prior article*

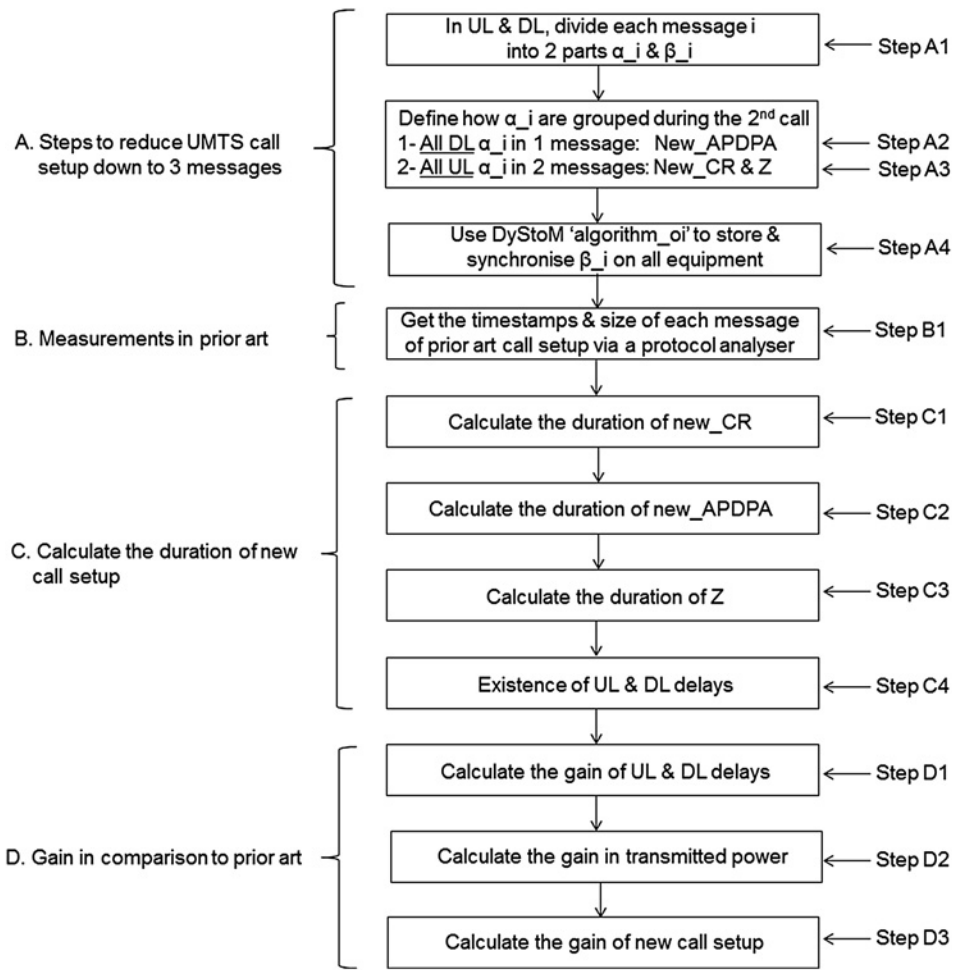

A. Steps to reduce UMTS call setup down to 3 messages
In UL & DL, divide each message i into 2 parts α_i & β_i
Step A1
Define how α_i are grouped during the 2nd call
1- All DL α_i in 1 message: New_APDPA
2- All UL α_i in 2 messages: New_CR & Z
Step A2
Step A3
Use DyStoM 'algorithm_oi' to store & synchronise β_i on all equipment
Step A4
B. Measurements in prior art
Get the timestamps & size of each message of prior art call setup via a protocol analyser
Step B1
C. Calculate the duration of new call setup
Calculate the duration of new_CR
Step C1
Calculate the duration of new_APDPA
Step C2
Calculate the duration of Z
Step C3
Existence of UL & DL delays
Step C4
D. Gain in comparison to prior art
Calculate the gain of UL & DL delays
Step D1
Calculate the gain in transmitted power
Step D2
Calculate the gain of new call setup
Step D3

Fig. 5 *Steps to reduce UMTS call setup down to three messages*

Step A3: Group all UL α_i in two messages: New_CR and Z. In prior articles and as shown in Fig. 4, there are seven messages exchanged in the UL direction. α_i and β_i are two of them: the connection request (CR) and the CM service request (CMSR) are shown, respectively in Tables 3 and 4. Whereas we consider the APDPR as being composed of only the β part, based on remarks in the previous step A2 this message is not included in the following calculation.

As the CR and CMSR are of a request type they are embedded in one UL message, which is called the new_CR message

$$\text{new\_CR} = \alpha\text{\_CR} + \alpha\text{\_CMSR} \tag{2}$$

For three other messages, CS complete (CSC), SMC complete (SMCC) and RBS complete (RBSC), all of them are already used as a response to one corresponding DL message sent in the new_APDPA. For example, RBSC is a response to the RBS sent in the new_APDPA. That is why they are combined in one UL message, called Z, that is sent after processing the new_APDPA. Note that for simplicity reasons and due to the fact that these messages are of a small size, it is considered that each of these three messages is composed of only the α_i part. It follows that

$$Z = \text{CSC\_priorart} + \text{SMCC\_priorart} + \text{RBSC\_priorart} \tag{3}$$

Table 1 Security mode command

α_SMC	message type
	RRC transaction identifier
	integrity check info
	integrity protection mode info
	security capability
	CN domain identity
	UE system specific security capability

Table 2 Activate PDP context accepted

α_APDPA	protocol discriminator
	transaction identifier
	message type
	negotiated LLC SAPI
	PDP address
	protocol configuration options
	negotiated QoS
	radio priority

Table 3 Connection request

α_CR	message type
	measurement results on RACH
	initial UE identity
β_CR	establishment cause
	protocol error indication

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)

Table 4 CM service request

α_CMSR	message type mobile identity ciphering key sequence number (3 bits)
β_CMSR	skip indicator protocol discriminator CM service type mobile station classmark

Note that the UL measurement report message could be sent either with the Z message or afterwards. In this paper and for simplicity purposes it is sent after, and that is why it is not considered in the calculations. In all cases, it could not be sent before the UE has obtained a dedicating signalling channel, meaning before the UE receives the new_APDPA. Fig. 6 shows the UMTS call flow with the proposed new method. Note that the node B application part [12] and RANAP messages remain the same as those described in prior articles.

Step A4: Use the DyStoM algorithm_oi to store and synchronise the β_i on all equipment. A storage algorithm is required in order to keep the β parts synchronised on the UE and the RNC in all situations, for example, if the UE goes out of coverage for a long period. Two tag parameters called the UE_cellx_tag and the RNC_cellx_tag are defined. In the case of a first call, the first tag is sent in the RRC CR and the second tag is sent in the APDPA, whereas in the second and following calls the first tag is sent in the new_CR and the second tag is sent in the new_APDPA. Keep in mind that all rules on the tag as described in Section 2 apply.

The algorithm works as follows.

If the UE_cellx_tag # RNC_cellx_tag, then the call setup is performed as in the prior articles and without any storage. Otherwise, the new call setup is performed where only one message is sent in the DL and two messages are sent in the UL.

After defining how the messages are sent with the DyStoM, the measurements and the calculations of their durations and of their processing time are described in the following sections.

3.2 *Measurements in prior art*

This section is composed of only one step B1 as shown in Fig. 5.

Step B1: Obtain the timestamps and the size of each message in prior articles via a protocol analyser. The size and the duration of each RRC message of an R99 PS call setup carrying a configuration of DL384/UL164 kbps are taken via a protocol analyser, inserted between the RNC and UE, and are shown in [8]. However, the timestamps of the messages is not shown in [8]. That is why another R89 PS call setup is performed instead, and the timestamps are shown in Fig. 7.

Only RRC messages are shown in Fig. 7 as RRC is the only protocol affected by the proposed method.

By calculating the difference of the timestamp of the last message activate PDP context accept with the timestamp of the first message 'CR', the duration of an UMTS call setup according to prior articles is

$$16:42:20:359 - 16:42:18:219 = 2140\,\text{ms}$$

3.3 *Calculate the duration of new call setup*

To calculate the duration of the new call setup, first the duration of each message, the new_CR, new_APDPA and Z message is calculated. Then an additional duration is added on top of them which is caused by a processing time and delays on the interfaces. This is shown in Fig. 8 and then the calculations are described in the following four steps: C1, C2, C3 and C4.

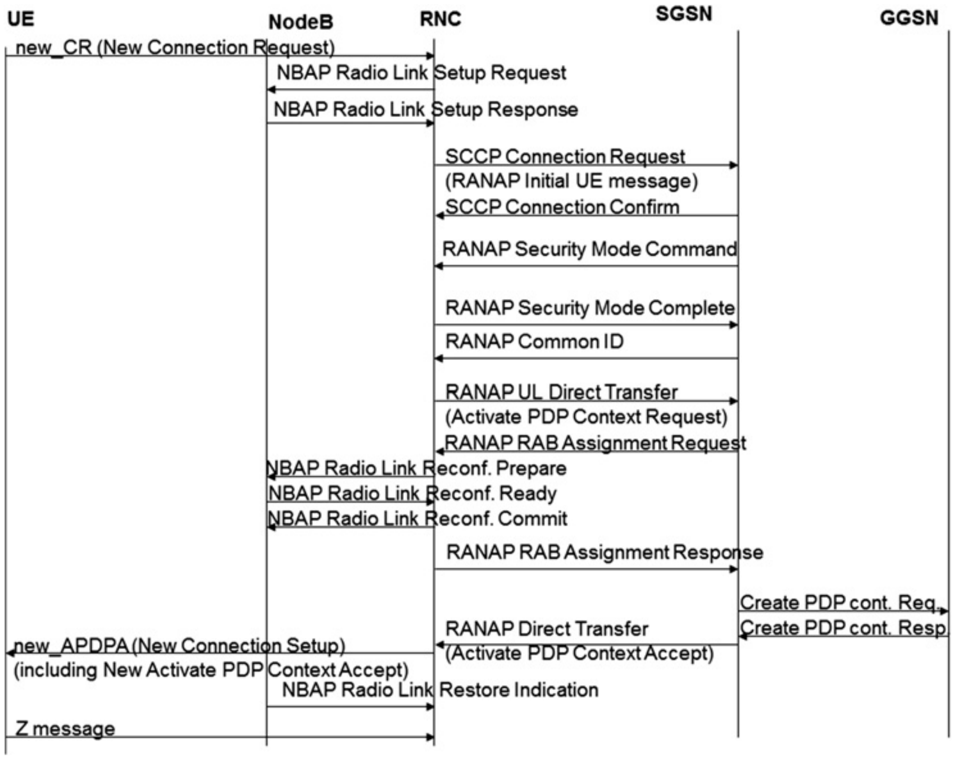

UE
NodeB
RNC
SGSN
GGSN
new_CR (New Connection Request)
NBAP Radio Link Setup Request
NBAP Radio Link Setup Response
SCCP Connection Request
(RANAP Initial UE message)
SCCP Connection Confirm
RANAP Security Mode Command
RANAP Security Mode Complete
RANAP Common ID
RANAP UL Direct Transfer
(Activate PDP Context Request)
RANAP RAB Assignment Request
NBAP Radio Link Reconf. Prepare
NBAP Radio Link Reconf. Ready
NBAP Radio Link Reconf. Commit
RANAP RAB Assignment Response
Create PDP cont. Req.
Create PDP cont. Resp
RANAP Direct Transfer
(Activate PDP Context Accept)
new_APDPA (New Connection Setup)
(including New Activate PDP Context Accept)
NBAP Radio Link Restore Indication
Z message

Fig. 6 *Proposed UMTS packet switch call setup*

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)

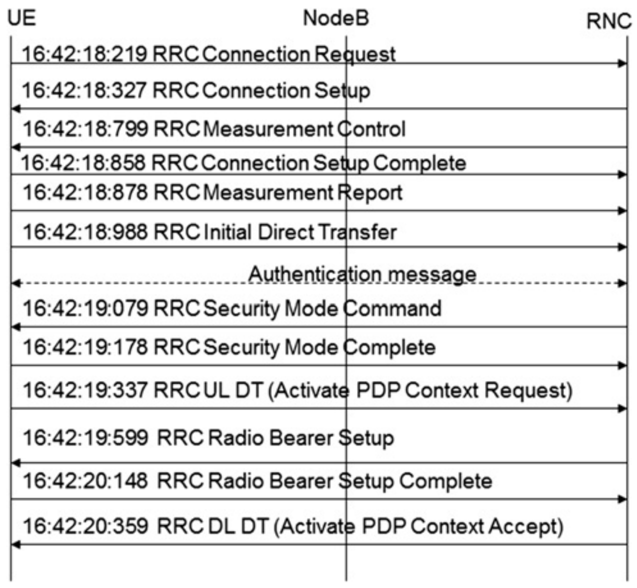

UE
NodeB
RNC
16:42:18:219 RRC Connection Request
16:42:18:327 RRC Connection Setup
16:42:18:799 RRC Measurement Control
16:42:18:858 RRC Connection Setup Complete
16:42:18:878 RRC Measurement Report
16:42:18:988 RRC Initial Direct Transfer
Authentication message
16:42:19:079 RRC Security Mode Command
16:42:19:178 RRC Security Mode Complete
16:42:19:337 RRC UL DT (Activate PDP Context Request)
16:42:19:599 RRC Radio Bearer Setup
16:42:20:148 RRC Radio Bearer Setup Complete
16:42:20:359 RRC DL DT (Activate PDP Context Accept)

Fig. 7 *Timestamps of UMTS PS call setup in prior articles*

It therefore follows that

$$\begin{aligned}\text{New\_call\_setup\_duration} &= \text{duration(new\_CR)}\\ &\quad + \text{UL processing delay} + \text{duration(new\_APDPA)}\\ &\quad + \text{DL processing delay} + \text{duration}(Z)\end{aligned} \tag{4}$$

By using the nominations in Fig. 8, it follows that

$$\text{New\_call\_setup\_duration} = t1 - t0 = T1 + T2 + T3 + T4 + T5 \tag{5}$$

Step C1: Calculate the duration of new_CR.

The CR is the first message that is sent during a call setup and is carried on the random access channel (RACH) channel. According to reference [13], during one single time transfer interval (TTI), equal to 10 or 20 ms depending on the UE selection, 168 bits of the signalling message are sent on an air interface. In this paper, the duration of the RACH is considered as 20 ms. The challenge to fit the new_CR (6) into the 168 bits still remains.

To calculate the size of the new_CR, the size of each IE listed in tables above is calculated.

α_CR = (‘Message Type’ + ‘Measurement results on RACH’ + Initial UE identity’) = 8 + 6 + 68 = 82 bits where Initial UE Identity is TMSI-and-LAI-GSM-MAP.

α_CMSR = (‘Message Type’ + ‘Mobile Identity’ + ‘Ciphering key sequence number’) = 6 + 48 + 3 = 57 bits.

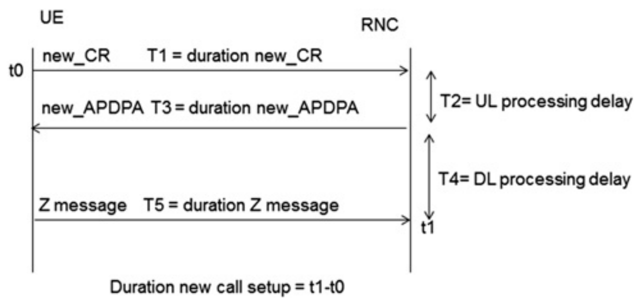

UE
RNC
t0
new_CR T1 = duration new_CR
new_APDPA T3 = duration new_APDPA
T2= UL processing delay
T4= DL processing delay
Z message T5 = duration Z message
t1
Duration new call setup = t1-t0

Fig. 8 *Duration and delays of the new call setup*

As a result, (2) becomes

$$\text{new CR} = 82 + 57 = 139\ \text{bits} \tag{6}$$

On top of 139 bits and 8 bits of the UE_cellx_tag, in addition another number of bits used as an overhead for Abstract Syntax Notation 1 (ASN.1) [14] encoding are added. This is left to vendor implementation. However, these additional bits should not exceed 168 – (139 + 8) = 21 bits.

Step C2: Calculate the duration of new_APDPA. In prior articles, the RRC CS is sent on the forward access channel (FACH) channel. Similarly, the new_APDPA which is a new version of CS is also sent on an FACH channel. In that case, according to reference [13], every TTI equal to 10 ms, two radio link control (RLC) [15] blocks are sent, with each composed of 152 bits.

As a consequence, the duration of the new_APDPA is calculated by dividing its size by 2 × 152 = 304 bits. For that purpose, and based on measurements and calculations performed in [8], in prior articles, the size of α_CS is 202 bits, the size of α_RBS is 278 bits, the size of SMC in prior articles is 120 bits and for APDPA the prior articles state 536 bits. It follows that

$$\text{new\_APDPA} = 202 + 278 + 120 + 536 = 1136\ \text{bits} \tag{7}$$

On top of the 1136 bits, 8 bits for the RNC_cellx_tag plus the additional ASN.1 overhead bits should be considered. Unlike the case of the new_CR, there is no size limit for the total size of the new_APDPA. Note that the size of all overheads is small in comparison with the size of the new_APDPA, and that is why it is not included in the following calculation. It follows that the number of TTI to carry new_APDPCA = 1136 bits/304 bits. This requires 4xTTI. It therefore follows that the duration of the new_APDPA is equal to 4 × 10 ms = 40 ms.

Step C3: Calculate the duration of Z message. In prior articles, according to [8], the size of the CSC is 208 bits, the size of the SMCC is 72 bits and the size of the RBSC is 56 bits. On the basis of (3), it follows that

$$Z = 208 + 72 + 56 = 336\ \text{bits} \tag{8}$$

All of these three messages are sent on an RLC AM mode [8]. As the Z message occurs after the RBS which is embedded in the new_APDPA, it follows that the Z message is sent on 3.4 kbps as described in [16], where according to [13] one RLC block carrying 128 bits is sent every TTI = 40 ms.

By dividing 336 bits by 128, it follows that the Z message requires 3 × TTI. The duration of the Z message is then 3 × TTI = 3 × 40 ms = 120 ms.

Step C4: Existence of UL and DL delays. Whenever a message is sent by either the RNC or by the UE, its response message from the opposite entity is not instantaneous. Instead there is a duration that is composed of the processing time in the equipment [17] and the delays on the interfaces. In this paper that duration is represented by the UL_delay (Resp–Req) for a delay at the RNC and by the DL_delay (Resp–Req) for a delay at the UE, where Resp is the response message and Req is the request message, and the delay would correspond to the difference of timestamps on the receiver side between the Resp timestamp and the Req timestamp.

3.4 Gain comparison with prior articles

This section is composed of three steps: D1, D2 and D3 as shown in Fig. 5.

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)

Step D1: Calculate the gain of the UL and DL delays.
Step D1.1: Case of UL: With the proposed new method, there is only one delay in the UL as there is only one message in the DL. That delay is represented by T2 in Fig. 8. It corresponds to the difference in the timestamp between the new_APDPA and the new_CR. For simplicity purposes and to be relevant to prior articles, the UL_delay (new_APDPA–new_CR) is considered to be equal to the delay in prior articles UL_delay (CS–CR). That consideration is because of two factors. First, the new_APDPA is sent on an FACH message which carries the CS in a prior article. Second, the delays at the core side required to gather the information from different messages to be sent in one message are considered to be negligible. This is due to the fact that on the core side there is no scheduled information as in the case of the air interface. On the basis of Fig. 7

$$\begin{aligned} \mathrm{T2} &= \mathrm{UL\_delay(new\_APDPA - new\_CR)} \\ &= \mathrm{UL\_delay(CS - CR)} = 16:42:18:327 \\ &\quad - 16:42:18:219 = 108\,\mathrm{ms} \end{aligned}$$

Note that for all other UL messages that are exchanged in prior articles, for example, UL CMS and UL APDPA messages, they trigger a RANAP message between the RNC and core network and their corresponding UL_delay is considered to be null in this paper. This is due to the fact that in prior articles, as shown in the figure, the delay of RANAP messages is very small.

As a conclusion, the new method does not bring any gain in UL processing delays in comparison with prior articles.

Step D1.2: Case of DL: In prior articles, in DL the biggest delays are the ones that occur during the signalling channel configuration DL_delay (CSC–CS) and then during the data channel configuration DL_delay (RBSC–RBS). From Fig. 7, their corresponding DL_delay are

$$\begin{aligned} \mathrm{DL\_delay(CSC - CS)} &= 16:42:18:858 \\ &\quad - 16:42:18:327 = 531\,\mathrm{ms} \end{aligned}$$

$$\begin{aligned} \mathrm{DL\_delay(RBSC - RBS)} &= 16:42:20:148 \\ &\quad - 16:42:19:599 = 549\,\mathrm{ms} \end{aligned}$$

Other delays, for example, the DL_delay (SMCC–SMC) are not counted as their values are very small. It follows that the total DL processing delays in prior articles is

$$\begin{aligned} &\mathrm{DL\_delay(RBSC - RBS)} + \mathrm{DL\_delay(CSC - CS)} \\ &= 531 + 549 = 1080\,\mathrm{ms} \end{aligned}$$

With the proposed method, the new_APDPA carries both the signalling and data channel configuration. As a result, the signalling and data channel processing are performed simultaneously, and as a consequence the DL_delay (new_APDPA–*Z*) in this paper is considered to be equal to the max (DL_delay (RBSC–RBS), DL_delay (CSC–CS)) = 549 ms.

It follows that the gain of the processing time in comparison with prior articles is

$$1080 - 549 = 531\,\mathrm{ms}$$

Step D2: Calculate the gain in transmitted power. In UMTS, every message is transmitted with a certain power, and regardless of whether it is in UL or in DL, it is transmitted with a certain power. The formulas used by the UE and by the Node B are vendor specific and they take a few factors into consideration, for example, the radio power path loss. In this paper, for the same radio conditions, for example, the same radio path, we consider that the same amount of radio energy is consumed with the proposed method and that in prior articles. The gain is then deduced by comparing the total transmitted duration with the proposed method and that described in prior articles. In prior articles, according to [8] the total duration of all RRC messages in DL is 590 ms, and in UL it is 660 ms. With the proposed new method, the total duration of all DL is represented by one message, the new_APDPA is 40 ms, and for UL it is new_CR (20 ms) + *Z* (120 ms) which is equal to 20 + 120 = 140 ms. It follows that the gain in transmission time in DL is 93% (40 out of 590 ms) and in UL it is 79% (140 out of 660 ms).
Step D3: Calculate the gain of new call setup. On the basis of (4), the duration of the call setup with the proposed new method becomes

$$\mathrm{New\_call\_setup\_duration} = 20 + 108 + 40 + 549 + 120 = 837\,\mathrm{ms} \tag{9}$$

Therefore the gain in comparison with prior articles is

$$2140 - 837 = 1303\,\mathrm{ms}$$

Note that in order to further reduce the call setup duration the UL_delay and DL_delay should be reduced. This requires an improvement in the actual hardware and software on all equipment involved in the UMTS call setup.

A summary of all gains is shown in Table 5.

Table 5 Summary of gains by the proposed method

Type of benefit	Prior art, ms	Proposed method, ms	Gain, %
DL processing delay	1080	549	50%
UL processing delay	108	108	0%
DL transmission duration	590	40	93%
UL transmission duration	660	140	79%
duration PS call setup	2140	837	61%

4 Conclusion and future work

In this paper, a novel generic storage method is described. It consists of reducing the size of signalling messages by storing repetitive information or repetitive tasks during a first call whenever certain conditions apply in order to avoid processing them in the following calls.

The application of this generic method to UMTS was described. It was shown that there is a huge benefit achieved by this application. In fact the UMTS call setup was reduced down to three messages, and as a consequence there is less call setup delay, less processing and less energy consumption in comparison with that described in prior articles, and by having less signalling messages exchanged and a shorter message size the probability of interference is reduced.

4.1 Future work

The same method could apply to many signalling procedures, regardless of whether in UMTS or in other systems such as wireless or wire line.

On the other hand, there is one case left where the existing generic method does not apply. It corresponds to the 'CR' case where the message to reduce is the first message that is sent from the server to the client. This is left for future study.

5 References

[1] 3GPP TS 25.331: 'UMTS; Radio Resource Control (RRC)', v11.8.0, 2014-01
[2] Grilli F., Vayanos A.: 'Default configurations with differential encoding in wireless communication system'. US Patent US20060040645, February 2006
[3] Mikola J.: 'Channel setup in a wireless communication network'. US patent 2005/0250504, November 2005
[4] Fischer P., Feuillette R.: 'Method and procedures for radio bearer setup'. US Patent 2010/0182963, July 2010
[5] Kitazoe M., Grilli F., Tenny N.E.: 'Stored radio bearer configurations for UMTS networks'. US Patent US20060229102, October 2006
[6] Gupta M., Koc A., Vannithamby R.: 'Reduced signaling overhead during radio resource control (RRC) state transitions'. Patent WO2013106060, July 2013
[7] Xu B., Ling L., Tao W.: 'Methods for air interface message transfer in fast call setup processes'. US Patent 7853258, December 2010
[8] Heik Y.B., Tafazolli R.: 'Dynamic storage method for the UMTS radio resource control'. Mosharaka for Researches and Studies, Fourth Int. Conf. on Communications, Computers and Applications (MIC-CCA 2011), 2011, pp. 49–53
[9] Heik Y.B., Tafazolli R.: 'Storing the RRC measurement control message in order to make earlier measurements'. Mosharaka for Researches and Studies MIC-CNIT2011, 2011, pp. 95–99
[10] Heik Y.B., Tafazolli R.: 'Reducing the size of two NBAP messages exchanged during a UMTS call setup'. Mosharaka for Researches and Studies MIC-CNIT2011, 2011, pp. 91–94
[11] 3GPP TS 25.413: 'UTRAN Iu interface RANAP (Radio Access Network Application Part) signaling', v11.5.0, 2013-12
[12] 3GPP TS 25.433: 'UMTS, UTRAN Iub interface Node B Application Part (NBAP) signaling', v11.7.0, 2014-01
[13] 3GPP TS 34.108: 'UMTS, LTE, Common test environments for User Equipment (UE), Conformance testing', v11.7.0, 2013-10
[14] ITU-T Rec. X.680-X.683: 'ASN.1 (Abstract Syntax Notation One)'
[15] 3GPP TS 25.322: 'UMTS, RLC (Radio Link Control)', v11.2.0, 2013-04
[16] Heik B., Tafazolli R.: 'Reducing the emergency call setup duration in UMTS'. Proc. IEEE 19th Telecommunications Forum TELFOR 2011 Serbia, Belgrade, November 2011, pp. 343–346
[17] Mishra A.: 'Performance characterization of signaling traffic in UMTS core networks'. IEEE Globecom, 2003, pp. 1141–1146

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)