Re-scheduling with temporal and operational resources for the mobile execution of dynamic UMTS applications

Roman Englert

Rheinische Friedrich-Wilhelms-Universität Bonn
Institute for Computer Science III
53117 Bonn, Germany
englert@iai.uni-bonn.de

Abstract. UMTS enables the execution of applications in mobile terminals. To start an application in a mobile terminal the UMTS call set-up is required. This procedure takes between a couple of seconds for an interactive game like chess and 30 seconds for WAP access. Often users start several applications and as a consequence the waiting period until the call set-ups are executed takes minutes. When a new application is started the waiting period for the execution of the already initiated applications can be reduced significantly by re-planning. In this situation the plan becomes incomplete since the new application is not considered. The concept of plan completion for incomplete plans with resources is introduced. Experiments for the initiation of up to 10 mobile applications demonstrate the improvement that is brought by this concept.

1 Introduction

The Universal Mobile Telecommunication Standard (UMTS) enables the implementation of various new applications like video telephony and online group games [10, 8]. The broad bandwidth based on the UMTS frequencies and the guaranteed quality of service compared to GSM make the execution of mobile applications possible [12]. Mobile applications are distributed in the mobile terminal (short: terminal) and the radio network. Several problems can occur during the call set-up (Fig. 1). Important examples are:

- incompatibility of the mobile with the WAP[1] gateway,
- memory violation of a Java application (J2ME: Java 2 micro edition), and
- insufficient resources of the radio bearer.

[1] Wireless application protocol

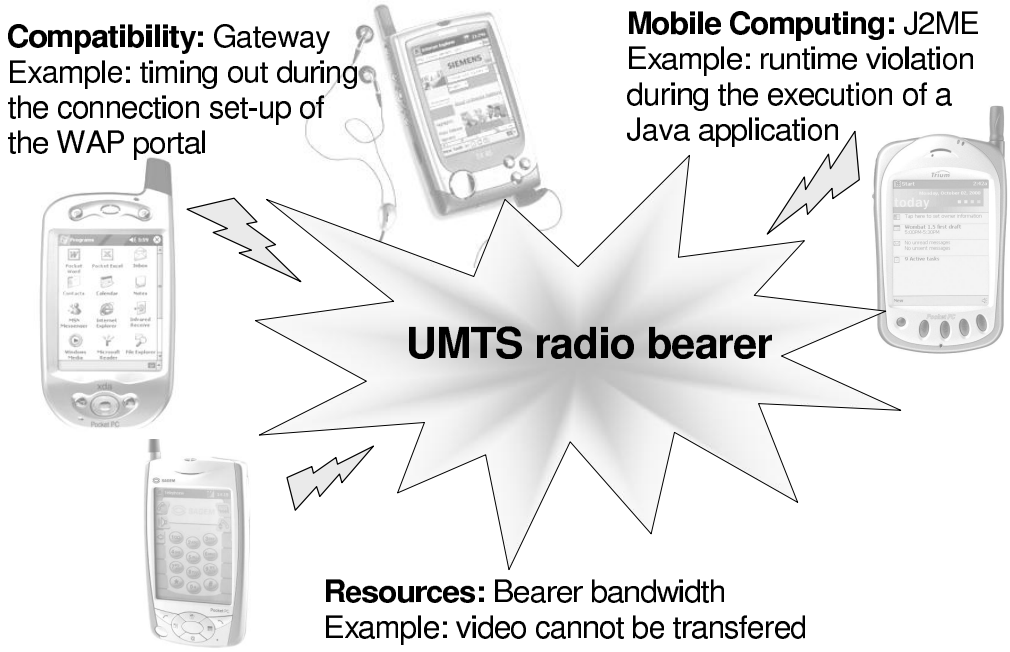

Fig. 1. Frequently occurring problems during the UMTS call set-up for the execution of mobile applications.

Whenever a problem occurs, the goal is to restart the call set-up with adapted parameters. Another challenge is the following: assume someone starts several applications on her/his mobile, e.g. a chess game and a news ticker for market prices. Then the goal is to execute both applications immediately after their initiation. Unfortunately, in most situations the radio bearer can only be used sequentially for the set-up of several calls from one terminal, and therefore, the execution of applications via the radio network requires patience on the user's side. Limited resources in mobile terminals and the radio bearer make a subsequent execution of mobile applications necessary. The examined hypothesis is that the call set-up for the execution of several mobile applications can be optimized by scheduling. As a result, mobile applications are executed immediately more or less in parallel after their start on the mobile. Thus, the waiting

period until all applications are started is reduced. Unfortunately, users start UMTS applications subsequently from one terminal and thus the applications have different starting points. Hence, for scheduling the world appears dynamic, since during planning a new application may be started and should be involved into the plan generation process. This situation is examined in this paper. In contrast to the dynamic world mobile applications in the static world have coincident starting points. This scenario has been investigated in [14]. As a result is shown that the UMTS call set-up for the execution of mobile applications in static worlds can be optimized by approximately 68%, i.e. if someone starts two mobile applications with coincident start points on one terminal then the waiting period until the call set-ups for the execution of both applications are performed is reduced by 68% in comparison to the situation where no scheduler is applied.

In this paper the optimization of call set-ups for the execution of mobile applications with different start points is investigated. The following section contains a brief description of the UMTS call set-up. In general, the call set-up using radio bearers is described based on the perspective of signaling from the field of electrical engineering. This continuous description cannot be applied to scheduling. Therefore the call set-up has been partitioned into modules by adopting the view of intelligent software agents [14]. The use of agents in telecommunication systems is widespread [15, 1]. Subsequently, the concept of incomplete plans to represent dynamic worlds is introduced (Sect. 3). The completion of an incomplete plan leads to the efficient generation of a (complete) plan. This concept is applied in Sect. 4 to various experiments with the multi-purpose planner TP4, that can handle temporal and operational resources like module execution time and radio bearer resources. The experiments examine the above described challenges of UMTS call set-ups (see begin of this section) with varying numbers of mobile applications. The conclusion is given in Sect. 5.

2 UMTS Call Set-up

In this section the UMTS radio bearer is described. The signaling flow in a radio bearer is a continuous process and scheduler need discrete actions. Hence, the call flow is modularized by adopting the perspective of *intelligent software agents* [8, 9].

Probably the best known feature of UMTS is higher bit rate [10]: packet-switched connections can reach up to 2 Mbps in the optimal case.

Compared to existing mobile networks, UMTS provides a new and important feature, namely negotiation of the *Quality of Service* (QoS) and of transfer properties. The attributes that define the characteristics of the transfer are throughput, transfer delay, and data error rate. UMTS bearers have to be generic to provide good support for existing applications and the evolution of new applications. Applications and services are divided into four traffic classes by their QoS [11, 10], where the traffic classes, their fundamental characteristics, and examples for applications are summarized in table 1.

Class	Conversational	Streaming	Interactive	Background
Constraints	Preserve time relation between information flow on the stream. Conversational pattern (stringent and low delay)	Preserve time relation between information entities of the stream	Request response pattern. Preserve data integrity	Undefined delay. Preserve data integrity
Examples	Voice, video telephony & video games	Streaming multimedia	Web browsing, network games	Background download of e-mails

Table 1. UMTS quality of service classes and their characteristics.

The main distinguishing factor between these classes is how delay-sensitive the traffic is: the conversational class is very delay sensitive (approximately 40 ms time preservation), and the background class has no defined maximum delay. In order to modularize the call set-up flow, the signaling of the bearer network is considered in the remainder of this section. It is assumed that mobile applications are executed in the terminal and that required data are provided via a *public domain network* (PDN) [12], e.g. the Internet. As an example consider a chess game, where player A and B start the application `chess` in their terminals and the moves are transfered via the radio network.

Networks for mobile data transfer can be divided into two units [13]: the *access network domain* (AND) and the *core network domain* (CND). The access from a mobile is done via the base station transceivers (BTS) or Node B in case of UMTS. Then the radio network controller (RNC) is setting up the connection and establishes the bearer service based on the CND resource availability. The CND contains units for the mobile data check (subscriber identification (HLR), authentication (AuC), and equip-

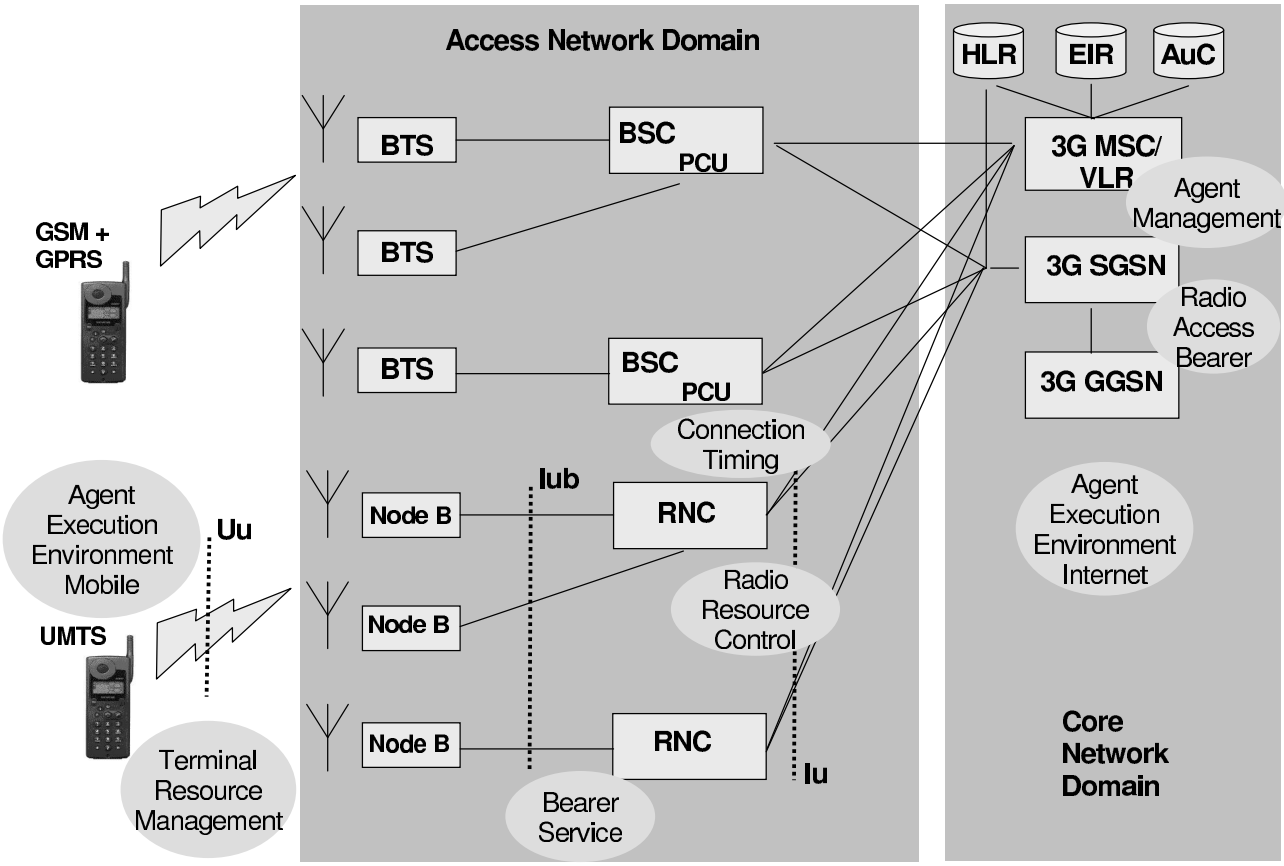

Fig. 2. UMTS radio network architecture.

ment registration (EIR)) that are controlled by the 3G mobile switching center (MSC). Access to a public domain network (PDN) like the Internet is provided via the 3G service and GPRS gateway support node (GGSN).

The UMTS call set-up can be modularized based on the perspective of *intelligent software agents* [8, 9], since agents are logical units and enable a discrete perspective of the continuous signaling process. The call set-up is partitioned into the following modules which are executed in sequential order [14] (see Fig. 2):

TRM Having initiated a mobile application (for the execution on a terminal) the resources of the terminal (display, keyboard, ...) are checked by the *terminal resource management* and allocated, if possible.

CT Transmission of "Ready for service" via the Node B to the terminal in order to ensure the *connection timing* for service availability.

AM Information of the terminal (location, id number, ...) is sent to AEEI. The transmission can be done comfortably by a so-called service agent [15] controlled by the *agent management* in the CND (Fig. 2). The advantage of a service agent is, that in case of failure, e.g. network resources are not available, the agent can negotiate with the terminal's agent about another QoS class (see Sect. 5).

AEEM Service agent with QoS class and parameters of application are sent from the mobile to AEEI.

RRC Provision of QoS by logical resources from the MAC level in the radio bearer's access network domain (Fig. 2).

RAB Radio bearer resources are supplied from the CND and the call flow is set-up.

AEEI Establishing data transfer from the core network to a PDN like the Internet and sending service agent (controlled by AM) to the application in the PDN to ensure the QoS for the application.

BS Establishing radio bearer resources with QoS and provide messages to the modules TRM and AEEI to start the execution of the application.

These modules are executed in sequential order to set-up a call for the execution of mobile applications. The problem fields that are depicted in the introduction are highlighted with respect to the modularized call set-up flow.

Compatibility of gateways, e.g. WAP gateway, with terminals is a challenge. Assume the modules TRM, CT, AM, AEEM, RAB, and RRC have been executed successfully. If the connection set-up to the WAP gateway takes too long, a timing-out occurs when the AEEI module is executed and the call set-up will be shut-down. In this case the mobile is assumed to be incompatible with the gateway.

Mobile Computing of a Java application can lead to an execution failure due to several reasons, e.g. 3D graphic output is not supported. Here the problem arises during the execution of the module AEEM. In this case the terminal capabilities have to be considered while selecting the fitting Java application via the AEEM. The selection can be done e.g. by negotiating the capabilities with the terminal's service agent. Afterwards the execution of the mobile application can be continued.

Resources of the bearer are limited, e.g. a demanded video cannot be transfered from the PDN to the mobile, since the required bandwidth

is not available anymore. Here the RRC determines the resource limitation of the radio bearer and a weaker QoS must be applied.

The implementation of the modules in the planning domain definition language (PDDL$^+$) [16] for scheduling is described in Sect. 4. Before, the concept of incomplete plans and their completion is introduced to enable the representation of mobile applications with different start points.

3 Incomplete Plans

A planning problem is given by an initial state (true and false facts), a goal (existence of true and false facts), and operators (set of descriptions of actions that contain names for possible actions, set of preconditions (atoms) to decide whether to apply the operator or not, and an effect). To change states operators are applied to states (sets of true and false facts, and actions with true preconditions). As result the effect describes how the situation changes (set of true and false facts).[2] The STRIPS-based planner TP4 [7] is applied to the experiments. STRIPS definitions for states and goals are the following [4]:

Definition 1. *A* STATE *is a conjunction of function-free ground literals, that is, predicates applied to constant symbols, possibly negated.*

Example for a state: $keyboard\_unlocked(terminal) \wedge not(keyboard\_used(terminal))$. States are complete, if an agent can obtain all literals in the environment. TP4 uses the convention that if a state description does not mention a given positive literal then the literal can be assumed as false.

Definition 2. *A* GOAL *is a conjunction of literals. The literals may contain variables that are assumed to be existentially quantified.*

TP4 restricts goals to conjunctions of atoms. Here, the scheduler asks for a sequence of actions that makes the goal true wrt. module execution durations and order constraints.

3.1 UMTS Call Set-up Scheduling within Static Worlds

In static worlds plans are assumed to be complete in the sense of a given order for the actions. In this context resources like answer delay and bearer resources are considered. These resources lead to the temporal ordering of actions:

[2] For a comprehensive description of planning problems see [3] and for current planning directions see [6].

Definition 3. *A* COMPLETE PLAN *is a temporal ordering of actions, where each variable of the actions is bound to a constant.*

Example: $\{trm(app_1) \wedge trm(app_2) \wedge ct(app_1) \wedge am(app_1) \wedge ct(app_2) \wedge am(app_2)\}$. Temporal ordering can be partially specified via the BEFORE constraint that is defined by the binary predicate $A\ BEFORE\ B$, with actions A and B:

Definition 4. *The* BEFORE CONSTRAINT *specifies actions A andB such that in any plan action A must be executed and finalized before action B (there may be other actions after A and before B).*

Example: $\{trm(app_1)\ BEFORE\ ct(app_1) \wedge (trm(app_1)\ BEFORE\ trm(app_2)\ \vee\ trm(app_2)\ BEFORE\ trm(app_1))\}$. Note, disjunction of constraints is allowed, too. In the above example the modules *trm* of both applications cannot be executed in parallel. For the UMTS call set-up two kinds of constraints occur:

Intra-application orders the modules of one application, e.g. $trm(app_1)\ BEFORE\ ct(app_1)$.

Inter-application orders modules with same names of different applications, e.g. $trm(app_1)\ BEFORE\ trm(app_2)$.

As scheduling goal the modules are to be ordered according to the ordering constraints and to the smallest execution duration of all applications.

3.2 Completion of Plans for Dynamic Worlds

The initiation of an application during the execution of a (complete) plan leads to the dynamic world scenario. As example consider the execution of a plan that contains data access via WAP. The call-set up for WAP takes approximately 30 seconds. If a mobile user wishes to start another application during this set-up phase, he/ she has to wait until the WAP access is completed. As a consequence, for scheduling the plan is incomplete due to state changes within this dynamic world scenario. The idea of incomplete plans is also described in [5].

Definition 5. *A* INCOMPLETE PLAN *is a set of actions whose temporal order may be incompletely specified by ordering constraints.*

Thus, an incomplete plan represents a finite set of complete plans. The set of ordering constraints that restricts an incomplete plan is a set of constraints, each of which consists of a conjunction of BEFORE constraints.

Example for an incomplete plan with the introduction of a new application during the execution of a plan (based on the example of a complete plan): $\{trm(app_1) \wedge trm(app_{new}) \wedge ct(app_{new}) \wedge am(app_{new}) \wedge trm(app_2) \wedge ct(app_1) \wedge am(app_1) \wedge ct(app_2) \wedge am(app_2)\}$. In this example app_{new} is started after module *trm* of application 1 is executed. As result several BEFORE constraints are violated, e.g. $ct(app_1) BEFORE\ ct(app_{new})$. Plan repair is done by the completion of incomplete plans:

Definition 6. *Assume the incomplete plan P is a finite set of complete plans $\mathcal{T}$. Then each complete plan $T \in \mathcal{T}$ is a* COMPLETION *of P, if T imposes the ordering constraints of P.*

Example: Let P be $\{trm(app1) \wedge trm(app2) \wedge ct(app1) \wedge am(app1) \wedge ct(app2) \wedge am(app2)\} \cup T' = \{trm(app_new) \wedge ct(app_new) \wedge am(app_new)\}$, where the union of plans means their concatenation with temporal order preservation. Then T' is a completion of P, since the BEFORE constraints can be fulfilled. Summarizing, partial states in incomplete plans refer to states in complete plans.

Definition 7. *Given an incomplete plan P with completions $\mathcal{T}$. The effect F of the executed action in the state S of P is true, iff F is true in the corresponding states S' of every completion $T \in \mathcal{T}$. Then S is called* COMPLETE STATE. *And S' of a completion $T' \in \mathcal{T}$ is* CORRESPONDING *to S of P, iff the effects F of S and F' of the executed action in state S' result in the same set of true and false facts.*

Hence, a complete state in an incomplete plan is a corresponding state in each completion. For example an effect F is true in state S_A in an incomplete plan P, if and only if F is true in the states in which action A is executed in every completion of P.

Now, given an incomplete plan the task is to find a completion. Execution start points for the new modules have to be found in such a manner that the waiting period is minimized for the user until all applications are executed. There are two kinds of BEFORE constraints: first, intra-application between the modules of one application $trm(\_)\ BEFORE\ ct(\_)$, etc., and second, inter-application between same modules of different modules, $trm(app_1)\ BEFORE\ trm(app_2)$, etc. For the completion of a plan by re-planning the inter-application constraints must be fulfilled according to the cost function defining that the summed-up execution time of all modules (resp. applications) is minimized and the order constraints hold. Note, further complexity is added, since same modules of different applications have different execution times, e.g. the execution time of module *trm* for a chess game (interactive QoS class) is shorter than *trm*

for a market ticker (streaming QoS class). The temporal scheduler needs to consider the above described constraints.

The trivial solution, i.e. consider two applications and execute each module as soon as possible wrt. the BEFORE constraints, does not lead to an optimal solution. Example: Let $P = \{trm(app_1, 30) \wedge trm(app_1, 20) \wedge ct(app_2, 25) \wedge am(app_1, 80) \wedge ct(app_2, 20) \wedge am(app_2, 40)\}$, where the second parameter denotes the execution time. Here it would be more efficient to execute $ct(app_2, 20)$ before $am(app_1, 80)$ since the execution of $am(app_2, 40)$ has to wait for the completion of $am(app_1, 80)$.

To implement the completion of an incomplete plan the already generated plan and the new mobile application are taken as input to find an appropriate completion that satisfies the constraints. In the case that the plan execution has started, the not yet executed modules only need to be considered in the re-scheduling process.

4 Implementation and Experiments

The domain is implemented in PDDL$^+$ [16] and applied to the heuristic planner TP4 [7]. TP4 is a planner with time based on heuristic search and enables the modeling of operational resources like data channels in the mobile. These features and specially the domain-independence resulted in the choice of TP4 from the currently available planners[3]. Fig. 2 in Sect. 2 depicts, that a mobile application can be executed, when the radio bearer with the required QoS is established. The predicate BS for the bearer establishment has as preconditions the successful execution of the module AEEI during the call set-up, the fulfillment of the required QoS class parameters (denoted as list L), and the transfered messages of the set-up status to the application in the mobile and the PDN (the complete list of predicates can be found in the appendix). The resources are already allocated by the preceding modules. Only the cost for the plan generation is considered, since having initiated an application the user is waiting for the plan generation and the call set-up including the plan execution. As effect the I_u bearer and the network connection for the mobile application are set up (Table 2).

The initiation of an application in the mobile starts with the execution of the module TRM (cf. Fig. 2 in Sect. 2). Afterwards, the module CT in the access network domain is asked for a ready-for-access signal. In the core of the call set-up is the radio access bearer procedure in the

[3] An overview of planning systems that handle resources like time or processors can be found in [2].

```
(:action BS
   :parameters (?A-new - application ?M - mobile ?L - list ?MSG1 - message)
   :precondition (and (aeei-ok ?A-new ?M ?L)
                      (qos-params ?A-new ?L)
                      (message-trm ?M MS1)
                      (message-aeei ?A-new MS2))
   :effect (and (iu-bearer ?A-new ?M ?L)
                (bs-ok ?A-new ?M ?L)))
```

Table 2. Bearer service establishment in PDDL$^+$.

CND. As first step the logical resources must be allocated (RRC), e.g. the required number of channels must be provided by the logical level in the radio bearer and later on these logical resources are mapped to physical channels (Table 3).

```
(:action RRC
  :parameters (?A-new - application ?M - mobile ?L - list)
  :precondition (and (ct-ok ?A-new ?M ?L)
                     (aeem-ok ?A-new ?M ?L))
  :resources ((logical-channels ?N : (required-channels ?A-new ?M ?N))
              (cell-update 1)
              (handover 1)
              (active-set-up 1))
  :effect (rrc-ok ?A-new ?M ?L))
```

Table 3. Radio resource controller in PDDL$^+$.

Finally, the bearer service is established and the demanded QoS class is provided to the mobile application. The UMTS call set-up domain has the following challenges for re-scheduling:

Real-time Can plans for the execution of mobile applications be repaired in an appropriate time? Plan repair has to be done with a maximum duration that does not exceed the original plan execution time.

Completeness Is it possible to repair the plan, i.e. does plan repair result in an (optimal) plan for the required applications that minimizes the waiting period until all applications are started?

The challenges described in the introduction cause a failure during execution (incompatibility of the mobile with the WAP gateway, memory violation during JAVA execution, and insufficient bearer resources). As

n applications	2	4	6	8	10
time [sec]	0.01	0.03	0.04	0.06	0.10

Table 4. Re-planning efforts in seconds for 2, 4, 6, 8, and 10 applications.

a consequence they require a restart with adapted parameters. We will focus on the lack in bearer resources since it may occur most frequent. Resources of the bearer are limited, e.g. a demanded video cannot be transfered from the PDN to the mobile, since the required bandwidth is not available anymore. Here the module RRC determines the resource limitation of the radio bearer and a weaker QoS must be applied. As experiments up to 10 applications are started with different time points. The UMTS call set-up domain as described in this section and Sect. 2 consists of the modules TRM, CT, AM, AEEM, RRC, RAB, AEEI, and BS, and additionally, the intra- and inter-application constraints. The constraints order the execution of the modules within an application and prevent the coincident execution of same modules.

It is assumed that a user initiates a new application in parallel with the execution start of these modules. Start point for re-planning of the experiments is the module AEEM. Let us consider an example. Assume a user has started a video application and the call set-up has already executed the modules TRM till AM from the computed plan. Then some effort for re-planning has to be spent. Afterwards the user starts another video application and the module RRC from the first call set-up is executed simultaneously with the module TRM for the new video application. Now re-planning is required, where as input the modules RRC till BS and the call set-up modules for the video application are applied. The same situation applies to the failure of a module execution and as a consequence a restart of the application is necessary.
As experiments the following is applied: assume a plan for the execution of n applications is computed. A user starts a new application during the execution of the plan. Table 4 shows the re-planning efforts for 2 up to 10 applications, where n means that one application is new and a plan for the call set-up of $n-1$ applications already exists. The required time for re-planning must be shorter than the execution of the plan, since as result of the plan execution the call set-ups are performed for the mobile applications. The execution of a call set-up takes between some seconds for an interactive game like chess [14] up to 30 seconds for a WAP access. Thus the re-planning effort is much smaller: it takes only between 0.01

and 0.2 seconds assuming that the instances for actions have been pre-computed as TP4 does. The good result demonstrates that a user reduces his waiting period by 50% until the execution of a new application starts, that is for one WAP access up to 30 seconds and for 10 applications up to 300 seconds, since mobile applications are executed sequentially.

5 Conclusion

The concept of plan completion for incomplete plans enables the modeling of dynamic mobile worlds, where applications are executed in mobile terminals. When a user starts a new application during the execution of a plan for mobile applications the current plan becomes incomplete. Plan completion is achieved by re-planning where the constraints for the UMTS call set-up modules must hold. The re-planning effort is much smaller than the plan execution time and thus the user's waiting period until all applications are started is significantly reduced by up to 300 seconds for 10 applications (WAP access).

In future re-planning for the negotiation of quality of service class parameters will be examined. If the execution of a video application fails due to lack in bearer resources a solution is to reduce the quality of service class parameters, e.g. reduce the size of the video frame.

References

1. Hayzelden, A., J. Bigham: Software Agents for Future Communication Systems. Springer, Berlin [1999]
2. www.ai-center.com/info/planningwithresources.html [2001]
3. Russell, S., Norvig, P.: Artificial Intelligence - A Modern Approach. Prentice Hall, U.S.A. [1995]
4. Blum, A., Furst, M.: Fast Planning Through Graph Analysis. Artificial Intelligence, Vol. 90, pp. 281 - 300 [1997]
5. Chien, S.: An Explanation-Based Learning Approach to Incremental Planning. Report UIUCDCS-R-90-1646, Dept. of CS, University of Illinois [1990]
6. Ghallab, M., Milani, A. (eds.): New Directions in AI Planning. Vol. 31 in Series Frontiers in Artificial Intelligence and Applications, IOS Press Omsha, Amsterdam [1996]
7. Haslum, P., Geffner, H.: Heuristic Planning with Time and Resources. Proceedings IJCAI Workshop on Planning with Resources [2001]
8. Appleby, S., Steward, T.: Mobile Software Agents for Control in Telecommunication Networks. Chapter 11 in: Hayzelden, A., Bigham, J. (eds.): Software Agents for Future Telecommunication Systems. Springer, Berlin, Tokyo [1999]
9. Busuioc, M.: Distributed Intelligent Agents - A Solution for the Management of Complex Telecommunications Services. Chapter 4 in: Hayzelden, A., Bigham, J. (eds.): Software Agents for Future Telecommunication Systems. Springer, Berlin, Tokyo [1999]

10. Holma, H., Toskala, A. (eds.): WCDMA for UMTS. Wiley & Sons, U.K. [2000]
11. 3^{rd} Generation Partnership Project: Technical Specification. Group Service and System Aspects: QoS Concept and Architecture. (Release 5), TS 23.107, V5.3.0, www.3gpp.org [2002
12. Kaaranen, H., Ahtiainen, A., Laitinen, L., Naghian, S., Niemi, I.: UMTS Networks: Architecture, Mobility and Services. Wiley, Chichester, U.K. [2001]
13. Walke, B.: Mobilfunknetze und ihre Protokolle. Teubner, Braunschweig, Germany [2000]
14. Englert, R., A. B. Cremers, A.B.: Configuration of Applications for the 3rd Generation Mobile Communication. In: Sauer, J. (ed.): Proceedings of the KI-2001 Workshop "AI in Planning, Scheduling, Configuration and Design". Vienna, Austria [2001]
15. Farjami, P., Görg, C., Bell, F.: Advanced Service Provisioning Based on Mobile Agents. Computer Communications. Special Issue: Mobile Software Agents for Telecommunication Applications [2001] 754 - 760
16. Fox, M. and Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. www.dur.ac.uk/maria.fox [2001]