MASTER

Distributed resource control for B-ISDN release 2/3

Moerkerken, J.

Award date:
1996

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
Master's Thesis:

Distributed Resource Control for B-ISDN Release 2/3

J. Moerkerken

Coach: ir. J.A. Peek (AT&T NS NL, Huizen)
Supervisor: Prof.ir. J. de Stigter
Period: April 1995 - January 1996

The Faculty of Electrical Engineering of Eindhoven University of Technology does not accept any responsibility regarding the contents of Master's Theses.
Summary

In the RACE project R2044, also known as the MAGIC project, an attempt was made to describe the signaling protocols for B-ISDN Release 2/3. In the applied specification methodology, a functional model for the network has been defined. This model consists of three functional layers: call (CC), resource (RC) and bearer control (BC), which together are responsible for all actions necessary to set up, maintain and release a call. The modeling of resource control, as presented in MAGIC, is not finished. Main target of this thesis is to complete the modeling for resource control, using the results of the MAGIC project as a starting point. These results are used as a guideline in the modeling of resource control, and modified if they induced limitations upon the modeling.

Resource control is responsible for allocating the resources, necessary for a service request, received from call control. In this thesis a framework is presented for resource control. In the framework algorithms perform the actual translation of a service request to physical resources. The quality of the translation is defined by three criteria: the translation time, the costs of the resources used and the costs of the connections made. For all three criteria a threshold value is introduced. The threshold values define which translations are acceptable, considering the criteria.

The framework consists of the following elements:

1. For the translation of the service request to physical resources, RC entities (independent functional elements, which can perform various functions on various controllable objects) require a certain knowledge of the resources, the user-locations and the layout of the network. For the modeling of this knowledge a network model is necessary. The network (domain) presented in this thesis is a collection of sub-networks (local domains). The local domains consist of nodes and endnodes. Local domains are connected to other local domains through interconnection trunks between the endnodes. The modeling of the knowledge of the RC entities can be solved using distributed platforms. In this thesis a model is presented as a reference model. In this model, each RC entity has knowledge of the local domain it belongs to, and the RC entities at the endnodes have the ability to obtain knowledge on other parts of the domain through a database system. The model of the network, as well as the model of the distributed RC knowledge, can be expanded for multi-level networks.

2. Interfaces have been enhanced for the communication between the functional entities in the network. The interface between call control and resource control is a black box description of the service, with the input/output relations of all parties involved. The format of the communication between peer RC entities depends on the knowledge of both RC entities. The interfaces have been updated to include the thresholds for the three criteria. Modifications of services are possible without interruption of the signal flow to users.

3. Depending on the knowledge of an RC entity and the format of the received service request, the action flow diagrams for three different functional RC entities can be described. One is for the RC entity receiving the service request from call control, one for the RC entities at endnodes and one for the other RC entities. The action flow diagram of an RC entity with a larger knowledge, receiving a service request with more freedom in the translation, encompasses the action flow diagram of an RC entity with less knowledge, receiving a service request with less freedom in the translation.

The framework and algorithms, performing the actual translation of a service request to resources, completely describe resource control. The strength of the framework is that the actual functionality of resource control is described by the algorithms, whereas the framework only supplies an environment in which the algorithms become operable. The definition of the optimal functionality of resource control is left open to the network operator, through implementation of the criteria in the algorithms. Better translation methods (algorithms) which may become available, can be implemented in the developed framework.

The model is ready to be implemented and tested. The implementation model should allow for modifications and enhancements to the algorithms and criteria. The implemented network model should allow for expansion of the model and freedom (for the network operator) in the implementation of the distributed knowledge of RC entities.
Preface

From April 4th, 1995 until January 31st, 1996, I have worked on this Master Thesis at the Advanced Engineering Research Concept Center (AER CC) of AT&T NS-NL, in Huizen, the Netherlands, under supervision of Alexander Peek. My external supervisor and graduation Professor at the Eindhoven University of Technology was Prof. ir. J. de Stigter.

After a rocky start, due to some personal problems, and the fighting back of mononucleosis, which developed somewhere between August and September and which is still not completely out of my system, I was able to finish the task, which I was assigned to, with very satisfying results. I am very happy to hear the model I developed and the results of my study will be embedded in the PLATINUM project, and that, when time comes, the model will be implemented.

There is many a person I am in debt to for making a contribution to this thesis, in whatsoever form. First of all, my gratitude goes out to Frans Vervuurt, who was my roommate for ten months and had a hard time taking on the conversation with me day after day. Luckily this holds for the both of us, so I have no fear that permanent damage has been done. Other people who definitely should be accredited for their contribution are (in no order of appearance): Dr. Hamza Ouibrahim, Sietse van der Gaast, Frank Baumann, Piet Schaafsma and Hessel Idzenga, all members of the working staff at the AER CC and Reinier Morra and Marcel van der Kraats, who, during the time I was active in Huizen, wrote their Master Thesis there as well. My thanks also go out to Rian van Gaalen, who, at the Eindhoven University of Technology, took care of many formalities, involved in the process of receiving that much desired title of Master of Electrical Engineering.
Special thanks go out to Prof. ir. J. (Hans) de Stigter for taking on the difficult task of being my external supervisor throughout the ten months I was active here and for being my graduation Professor. But most of all I would like to thank Alexander Peek. Even though this was his first time, I can honestly say (after multiple experiences with others) that Alexander is an absolutely superb supervisor (and a great person as well, which always helps). I have found great joy in working under the supervision of him, and even though he was my superior, I always had the feeling we took on the task together and were each other’s equals throughout the process. Alexander was very hungry for results and always came up with the right questions at the right time. It has been a great pleasure working with him and I am sure if it weren’t for his patience, his hunger for results and his personality, I wouldn’t have been able to produce the thesis which you are now about to read.

Jochem Moerkerken,
Huizen, January 1996.
# Index

1. Introduction ................................................................. 1
   1.1. Introduction to B-ISDN ........................................... 1
   1.1.1. B-ISDN and ATM .................................................. 1
   1.1.2. B-ISDN Protocol Reference Model (PRM) ..................... 2
   1.2. Targets of this Thesis ........................................... 3
   1.3. Outline of the Report ............................................ 5

2. MAGIC Specification Methodology ....................................... 6
   2.1. Introduction ....................................................... 6
   2.2. Stage 1: Definition of the Services from the User's Viewpoint .............. 6
      2.2.1. Step 1.1: Prose Description .................................... 7
      2.2.2. Step 1.2: Static Service Description ............................ 7
      2.2.3. Step 1.3: Dynamic Description ................................... 8
   2.3. Stage 2: Identification of the Network Functional Elements and the Information Flows between Them ........................................ 10
   2.4. Stage 3: The Detailed Description of the Signaling Protocols ............. 12
   2.5. Conclusions ....................................................... 12

3. The MAGIC Functional Model ............................................. 13
   3.1. Introduction ....................................................... 13
   3.2. Layout of the MAGIC Functional Model .......................... 13
      3.2.1. Call Control ...................................................... 14
      3.2.2. Resource Control ................................................ 15
      3.2.3. Bearer Control .................................................. 15
   3.3. Global Modeling Overview ...................................... 15
   3.4. Explicit versus Implicit Model ................................... 16
   3.5. Local View Model ................................................. 17
      3.5.1. Resource Control Local View .................................. 18
   3.6. Conclusions ....................................................... 18

4. Criteria for Resource Control .......................................... 20
   4.1. Introduction ....................................................... 20
   4.2. Call Implementation Criteria .................................... 20
   4.3. Using the Criteria for Resource Control: Definition of the Variables ........ 22
   4.4. Definition of the Thresholds ...................................... 22
   4.5. Thresholds through the Implementation Phase of a Call ................... 25
   4.6. Conclusions ....................................................... 26

5. RC Network Modeling ..................................................... 27
   5.1. Introduction ....................................................... 27
   5.2. Domain Topology .................................................. 27
      5.2.1. Terminology ..................................................... 28
5.2.2. Validation of the Model ................................................................. 29
5.3. Example Distributed Platform for Distributed Resource Control Knowledge. 30
5.4. Conclusions.......................................................................................... 31

6. The Functional Model Interfaces......................................................... 33
   6.1. Introduction .................................................................................. 33
   6.2. General Remarks towards Modifications to the MAGIC Model ........ 34
   6.3. Entity Relations in One Stage Expressed in Numbers ...................... 35
       6.3.1. CCc Viewpoint ................................................................. 35
       6.3.2. RCc Viewpoint ................................................................. 35
       6.3.3. BCc Viewpoint................................................................. 36
   6.4. The CC-RC Interface .................................................................... 36
       6.4.1. Modifications to the MAGIC Model .................................... 36
       6.4.2. Mapping Rules for the CC-RC Interface ......................... 38
       6.4.3. The Objects of the CC-RC Interface ................................. 38
       6.4.4. Additional CC-RC Interface Information Flow .................. 39
       6.4.5. Syntax of the CC-RC Interface ........................................ 39
   6.5. Information Flow of the RC-RC Interface ..................................... 40
       6.5.1. Modifications to the MAGIC Model .................................... 40
       6.5.2. The Objects of the RC-RC Interface ..................................... 41
       6.5.3. Additional Information ...................................................... 44
       6.5.4. Syntax of the RC-RC Interface ........................................ 44
   6.6. The RC-BC Interface .................................................................... 44
   6.7. Conclusions.................................................................................... 45

7. Action Flow Diagrams for Resource Control....................................... 46
   7.1. Introduction .................................................................................. 46
   7.2. Terminology ................................................................................ 46
   7.3. General remarks .......................................................................... 48
       7.3.1. Remarks on RC Entities at Endnodes ............................... 48
       7.3.2. Remarks on Thresholds ..................................................... 49
       7.3.3. Remarks on the Actions Sequences .................................. 49
   7.4. Description of the Action Flow Diagrams .................................... 50
       7.4.1. RC Entity at Non-OLEX and Non-Endnode ....................... 51
       7.4.2. RC Entity at the OLEX ..................................................... 54
       7.4.3. RC Entity at Endnode ....................................................... 60
   7.5. Special Remarks on Modifications .............................................. 64
   7.6. Special Remarks on Releases ....................................................... 70
   7.7. Conclusions.................................................................................... 70

8. Using the Criteria in the Algorithms for Resource Control.................. 72
   8.1. Introduction .................................................................................. 72
   8.2. Introduction of the Algorithms: Using the Criteria for Optimizing Resource Control ......................................................... 72
   8.3. Conclusions.................................................................................... 74

9. Conclusions........................................................................................... 75
   9.1. Conclusions.................................................................................... 75
   9.2. Modifications to the MAGIC Model ............................................ 77
   9.3. Recommendations ....................................................................... 78
References ............................................................................................................ 80
List of Acronyms ................................................................................................. 83
List of Words ...................................................................................................... 85
Appendix 1. Call Control Local View ................................................................. 89
  Appendix 1.1. Elementary Call Objects .......................................................... 89
    Appendix 1.1.1. Control Modes ................................................................. 89
  Appendix 1.2. Call Associated Objects .......................................................... 90
  Appendix 1.3. Telecommunications Associated Objects ............................... 90
  Appendix 1.4. Channel Associated Objects ................................................... 90
  Appendix 1.5. Example Call Control Global and Local View Model ............. 90
Appendix 2. Bearer Control Local View ............................................................. 93
Appendix 3. Action Flow Diagrams for Generic Information Flows for Call Control ............................................................................................................ 94
  Appendix 3.1. Introduction ............................................................................ 94
  Appendix 3.2. Atomic Actions Concepts of CCR .......................................... 94
  Appendix 3.3. CCR Based Information Flows .............................................. 94
Appendix 4. Distributed Resource Control Knowledge ..................................... 96
  Appendix 4.1. Introduction ............................................................................ 96
  Appendix 4.2. Modeling Distributed Resource Control Knowledge ................ 97
    Appendix 4.2.1. Modeling Not Related to the Domain Topology ................. 97
      Appendix 4.2.1.1. Model 1: Totally Distributed Resource Control ............ 97
    Appendix 4.2.2. Distributed Resource Control within the Domain Topology ... 98
      Appendix 4.2.2.1. Model 2: Local Group Knowledge .............................. 98
      Appendix 4.2.2.2. Model 3: Global Access to Resource Database .......... 100
      Appendix 4.2.2.3. Model 4: Limited Access to Resource Database .......... 103
      Appendix 4.2.2.4. Model 5: All Endnodes Having Global Information ... 106
    Appendix 4.2.3. Alternative Models ......................................................... 108
      Appendix 4.2.3.1. Alternative Models within the Domain Topology .......... 108
      Appendix 4.2.3.2. Alternative Models Not Related to the Chosen Domain Topology ............................................................................................................ 109
  Appendix 4.3. Evaluation of the Alternative Models ....................................... 109
    Appendix 4.3.1. Data Management ............................................................ 109
      Appendix 4.3.1.1. Model 1 .................................................................. 110
      Appendix 4.3.1.2. Model 2 .................................................................. 110
      Appendix 4.3.1.3. Model 3 .................................................................. 111
      Appendix 4.3.1.4. Model 4 .................................................................. 111
      Appendix 4.3.1.5. Model 5 .................................................................. 112
    Appendix 4.3.1.6. Conclusions ............................................................... 112
  Appendix 4.3.2. Extra Hardware Requirements for Connections within the Domain ............................................................................................................ 112
    Appendix 4.3.2.1. Model 1 .................................................................. 113
    Appendix 4.3.2.2. Model 2 .................................................................. 113
    Appendix 4.3.2.3. Model 3 .................................................................. 113
    Appendix 4.3.2.4. Model 4 .................................................................. 114
List of Figures

Figure 1.1: B-ISDN protocol reference model (PRM) ..................................................... 2
Figure 1.1: Layered structure of the OSI reference model .............................................. 3
Figure 2.1: The MAGIC service description framework (SDF) ....................................... 8
Figure 2.2: The MAGIC dynamic service description model ............................................ 9
Figure 2.3: Different Party Type descriptions for the service ......................................... 10
Figure 2.4: Relationship between step 1.3 flows and stage 2 flows................................. 10
Figure 2.5: The MAGIC functional model ...................................................................... 11
Figure 3.1: The MAGIC functional model ...................................................................... 14
Figure 3.2: The MAGIC global modeling overview ....................................................... 16
Figure 5.1: Domain topology ......................................................................................... 28
Figure 5.2: Example of user groups .............................................................................. 29
Figure 5.3: Layered expansion of the network ................................................................ 30
Figure 5.4: Implementation example - limited access to resource database .................. 31
Figure 6.1: The stage 2 model with the resource control environment ......................... 34
Figure 6.2: Action sequence for the queries between RC entities ................................... 44
Figure 7.1: AFD for RC entity at non-OLEX and non-endnode ....................................... 52
Figure 7.2: Atomic action tree ....................................................................................... 54
Figure 7.3: Part of the AFD for RC entity at OLEX ....................................................... 55
Figure 7.4: AFD for RC entity at OLEX ........................................................................ 59
Figure 7.5: Atomic action trees ...................................................................................... 60
Figure 7.6: AFD for RC entity at endnode ...................................................................... 61
Figure 7.7: Atomic action trees ...................................................................................... 64
Figure 7.8: Addition of a new combiner for local calls .................................................. 65
Figure 7.9: Deletion of a combiner for local calls ............................................................ 66
Figure 7.10: Addition of a new distributor for local calls ............................................... 66
Figure 7.11: Deletion of a distributor for local calls ....................................................... 67
Figure 7.12: Addition of a new combiner for non-local calls .......................................... 67
Figure 7.13: Deletion of a combiner for non-local calls .................................................. 68
Figure 7.14: Addition of a new distributor for non-local calls ........................................ 68
Figure 7.15: Deletion of a distributor for non-local calls ................................................ 69
Figure 7.16: Allocation of a new combiner for non-local calls ........................................ 69
1. Introduction

1.1. Introduction to B-ISDN

1.1.1. B-ISDN and ATM

In the fast changing world of telecommunications today, the basis for future communications will be the concept of B-ISDN (Broadband Integrated Services Digital Network, [STA94]). B-ISDN will offer an extensive range of multimedia, multiparty services which should comply with the current and future needs of users. B-ISDN is a revolutionary new concept, based on the asynchronous transfer mode (ATM).

The term transfer in ATM comprises both transmission and switching aspects, so a transfer mode is a specific way of transporting and switching information in a network. ATM is based on a cell-by-cell transport mechanism. The use of short cells (packets of 53 bytes) in ATM and the high transfer rates involved result in transfer delays and delay variations which are sufficiently small to enable it to be applied in a wide range of services, including real-time services such as voice and video.

The architecture of the B-ISDN is described in [CCI327]. According to this recommendation, the information transfer and signaling capabilities of B-ISDN comprise:

- broadband capabilities
- 64 kbit/s ISDN capabilities
- user-to-network signaling
- inter-exchange signaling
- user-to-user signaling

In B-ISDN signaling channels are logically separated from the user channels (out-of-band signaling). However, the signaling information can be transported over the same media as the user information.

Signaling information is necessary for the setup, maintenance and release of connections between users, in a call. Old signaling sets do not meet the requirements for B-ISDN Release 2/3. New signaling requirements result from the more powerful service concepts that have become feasible in B-ISDN Release 2/3, such as multi-connection, multiparty calls. The new signaling demands are such, that a thorough study of signaling capabilities and requirements for B-ISDN is required. A layered approach of all system functions can be helpful (or essential) in the design. The B-ISDN protocol reference model (PRM) serves this purpose.
1.1.2. B-ISDN Protocol Reference Model (PRM)

In all telecommunication systems, a layered approach is used for the organization of all telecommunication functions. The functions of the layers and the relationships of the layers with each other are described in a protocol reference model (PRM). A description of the PRM for the existing ISDN is given in [CCI320]. In particular, it has introduced separate planes for the segregation of user, control and management functions (see Figure 1.1). This PRM is the basis for the B-ISDN PRM which is described in [CCI321].

![Figure 1.1: B-ISDN protocol reference model (PRM).](image)

In Figure 1.1 The B-ISDN PRM is depicted. It consists of three lower layers and a combination of higher layers. All layers can be mapped onto the OSI reference model ([CCX200]). Moreover, the layers are divided into three planes:

- the user plane, which provides for the transfer of user information.
- the control plane, which is responsible for call and connection control functions. These are the signaling functions, necessary for the set-up, supervision and release of calls.
- the management plane, which consists of two elements. The first element is plane management, which involves management functions related to the whole system. The second element is layer management, which performs management functions relating to resources and parameters residing in its protocol entities.

According to the open system interconnection (OSI) reference model of the ISO each open system can be described as a set of subsystems arranged in vertical sequence (see Figure 1.1).

---

1 Entity: An entity is an independent functional element, which can perform various functions on various controllable objects. Functional entities can be created and deleted and communicate with higher level, lower level and peer entities. Functional entities can create lower level and peer entities.
An $N$-subsystem can only interact with its peers or subsystems $(N-I)$ and $(N+I)$. For the communication between $N$-subsystem peers a $N$-to-$N$ protocol is used. Services of layer $N$ are provided to layer $(N+I)$, at the $N$-service access point ($N$-SAP).

### 1.2. Targets of this Thesis

For the description of the signaling protocols for B-ISDN Release 2/3, RACE\textsuperscript{2} project R 2044 (MAGIC) adopted the three staged methodology described in [CCI130]. In the second stage in this approach - the identification of the network functional elements and the information flows between them- a functional model for the B-ISDN network signaling entities is described. This model consists of three functional layers: the call and bearer control layers (Q.73 model equivalent ([CCQ73])), and the newly introduced resource control layer.

The responsibility of call control is to coordinate and manage the negotiation of telecommunication services amongst a set of users. It maps the objects of the call model onto a more concrete service request, which is passed on to resource control. If resource control has all the information about the requested service, the resource control service description can be translated into physical resources. Bearer control makes the connections, necessary between the various users and resources, involved in the service.

The modeling is not complete. Call control and bearer control functionality can be copied directly from Q.73 (refer to [CCQ73]), and modifications only have to be introduced if necessary for the increased functionality both control layers have to offer within B-ISDN Release 2/3. Resource control had not been described previously. In the MAGIC project, a model for resource control has been developed. The description of its functionality leaves many questions unanswered:

- What if an RC entity does not have all the information necessary about a service request?
- How is the information of RC entities modeled?

\textsuperscript{2} RACE (Research and Development on Advanced Communication Technologies in Europe) was a EC program which promotes research on advanced communications in Europe.
What rules should resource control apply in the translation of a service request into physical resources?

What information should be contained in the service request, which call control passes on to resource control?

Clearly the model is not finished. Besides, in the process of the modeling, no attention is paid to the criteria which play a role in the resource control functionality. The RC model has only been examined and developed as an element in the description methodology (applied in the MAGIC project) for developing the signaling protocols for B-ISDN Release 2/3.

The main target of this thesis is therefore the completion (and enhancement) of the resource control modeling. The main question to be solved is:

How does resource control translate a service request, received from call control, into physical resources?

This question can be divided into the following sub-questions:

• Which criteria play a role in the translation of a service request into physical resources and how are these criteria implemented in the RC functionality?
• How is the knowledge of resource control, necessary in the decision making in the translation of a service request to physical resources.
• How is this knowledge distributed throughout the network?
• How is a service request from call control modeled?
• How is the functionality of resource control distributed over the network, and how is the step-by-step approach used in the translation of a service request into physical resources?
• How is the actual translation performed by resource control?

In this thesis a framework is presented for implementing resource control functionality. The framework provides resource control with all the elements, necessary for the translation of a service request. For the solution of the problem the following staged approach is used:

1. Investigate the RC model, as introduced in the MAGIC project.
2. Definition of the criteria for resource control.
3. Description of the environment: definition of the layout of the network, the span of knowledge of each of the RC entities and the service request format from call control.
4. Description of the RC-RC interface.
5. Definition of the step-by-step approach of each of the (functionally separate) RC entities.
6. Development of algorithms for resource control, implementing the criteria.
7. Testing of the RC model.

Steps 1-6 are described in the remainder of this volume. Refer to [MOE96], for a large number of testruns (step 7). Step 3-5 represent the construction of the framework. In step 6 the actual translation of a service request is investigated. In this translation the criteria, defined in step 1, are used in the algorithms.

The work is part of the PLATINUM project. The PLATINUM project consists of four members: AT&T, Telematica Research Center (TRC), Deutsche Telekom and the Center for
Telematics and Information Technology (CTIT) of the Twente University of Technology. The project is sponsored by the ministry of Economic Affairs. The project adopted many concepts and ideas of the RACE R2044 (MAGIC) project which ultimate goal was to design signaling protocols for a wide range of multimedia and multiparty calls.

1.3. Outline of the Report

Chapter 2 explains the specification methodology used in MAGIC for the description of the B-ISDN Release 2/3 signaling protocols. In chapter 3 the functional model, from stage 2 of the specification methodology, is described. The functionality of each of the functional separate levels is described in detail.

In chapter 4 the criteria, playing a role in the definition of the optimization of resource control functionality, are defined. The criteria are evaluated in the actual translation phase of a service request by resource control. They are also taken into account in the definition of the framework.

In chapter 5 a model of the network and a possible distributed platform for the distribution of resource control knowledge are depicted. More examples of distributed platform solutions for the distribution of resource control knowledge can be found in Appendix 4, along with a comparison using various criteria.

Chapter 6 covers the interface definitions within the MAGIC functional model, along with the modifications upon the MAGIC definitions. In Appendices 5 and 6 the interface syntax and the ASN.1 descriptions of information flows are described for both the CC-RC and RC-RC interface. The RC-BC interface description is an open point which is to be solved in the PLATINUM project.

In chapter 7 the action flow diagrams for the various RC entities are described. The action flow diagram describe, step-by-step, all actions an RC entity takes in the translation of a service request. In this chapter all major modifications to the original MAGIC model, as well as all additions to the model, are given.

The previous chapters describe the framework for resource control. In the framework resource control can perform the actual translation of a service request using algorithms. Chapter 8 explains how the criteria, found in chapter 3, can be implemented within the algorithms resource control uses in the translation of a service request. In Appendix 9 a large number of possible algorithms are presented.
2. MAGIC Specification Methodology

MAGIC adapted the CCITT I.130 recommendations for describing the signaling protocols for B-ISDN Release 2/3. This methodology comprises three steps. In step 1 a large number of services are described from the user's point of view. For the static service description a service description framework (SDF) is developed. In the SDF, the service attributes are classified in levels. The dynamic service description is described at the user application level, describing the interactions between the various party types. In step 2 - the specification of the network functional elements and the information flows among them - a functional model is derived. The model consists of 3 layers of control: call (CC), resource (RC) and bearer control (BC). Protocols were defined in stage 3 for the interface between the user (terminal) and the network. Large parts of this chapter are copied from [MAG03].

2.1. Introduction

In 1991 RACE started project R2044, better known as MAGIC (Multiservice Applications Governing Integrated Control), in order to develop the signaling protocols for B-ISDN release 2/3. The signaling protocols for B-ISDN release 2/3 should be capable of handling complex multimedia, multiconnection, multiparty, standardized services and open services. MAGIC followed the methodology of CCITT Recommendation I.130 ([CCI130]) in developing these signaling protocols. This method comprises three sequential stages:

1. Definition of the services from the user’s viewpoint.
2. Identification of the network functional elements and the information flows among them.
3. Detailed description of the signaling protocols.

In this chapter the three stages will be discussed shortly to give the reader an overview of the MAGIC project specification methodology. Large parts of this chapter are copied from [MAG03], where more information can be found.

2.2. Stage 1: Definition of the Services from the User’s Viewpoint

The objective of the stage 1 description is to identify the overall service descriptions from the user's (terminal's) point of view. However, it does not deal with the details of the human interface itself. The CCITT I.210 Description method ([CCI210]) was used by the MAGIC project as a basis for the prose description method. The CCITT framework for stage 1 is split into three separate steps:

3 CCITT: International Telegraph and Telephone Consultative Committee.
• **STEP 1.1**: prose description;
• **STEP 1.2**: static description;
• **STEP 1.3**: dynamic description.

MAGIC adopted step 1.1 unchanged from the CCITT. However, steps 1.2 and 1.3 have been adapted by MAGIC, while still trying to remain as close to the CCITT approach as possible. For each of the steps a short overview of the results are given.

### 2.2.1. Step 1.1: Prose Description

In this step a prose description is given of the services to be offered by B-ISDN. The results can be found in [MAG03]. Services that are to be supported range from simple point-to-point telephone connections to complex multiparty, multimedia open services.

### 2.2.2. Step 1.2: Static Service Description

MAGIC has categorized the static service description attributes in a more comprehensive way by introducing the MAGIC service description framework (SDF, explanation follows shortly). Also, additional attributes have been introduced where required.

The current CCITT method for service description, as in Recommendation I.210 ([CCI210]), was originally defined for the use in the N-ISDN. As such, it is deficient when applied to the B-ISDN, in particular for the multimedia, multiparty, and multiconnection aspects of the B-ISDN ([STA94]).

MAGIC has therefore chosen to apply a more modular approach to service descriptions, classifying the attributes of a service in a number of levels. Each level is represented by a standardized table of attributes. This approach is termed the MAGIC service description framework (SDF, see Figure 2.1). The use of such a framework makes it possible to describe complex services using a number of simpler re-usable service elements. The framework also provides the means to describe the different types of parties in a multiparty service.
The attributes of the framework are:

- **TELECOMMUNICATION SERVICE LEVEL (TCS):** This is a name identifying the overall telecommunication service.
- **PARTY TYPE (PT):** The party type is used to describe the different behavior of various types of parties that may be involved in a call. Each party type may have a different configuration of USM's.
- **USER SERVICE MODULE (USM):** An USM is a basic building block of service. It provides a typing function, combining one or more information types of the service into a single grouping that is understandable to the user's view of the service (e.g. television).
- **PARTY EDGE (PE):** A PE is used to define the relationships between the PT's that may take part in a service, and the USM's that comprise the service.
- **ABSTRACT SERVICE MODULE (ASM):** The ASM encapsulates the user's view of the basic information types that may be used within a multimedia service (e.g. video, audio, data).
- **ACTUAL ASSOCIATION (AA):** The AA describes the relation between the ASM's and the PT's. This link was added to the framework in a later stage. It is necessary that there is a direct link between the ASM's and PT's. It is for example possible that a PT has an edge to an USM video, but he is not able to receive audio and therefore there is no AA to the ASM audio (which an object of the USM video).
- **SERVICE MODULE (SM):** The SM represents the PRM protocols used to implement a basic service type of a multimedia service.
- **ACCESS CONNECTION ELEMENT (ACE):** The ACE encompasses attributes of the B-ISDN bearer connections.

### 2.2.3. Step 1.3: Dynamic Description

4 The peer-to-peer layer protocols in the protocol reference model, see chapter 1.
In the MAGIC project it has been decided to define the dynamic service description at the user application level, describing the interactions between the various party types and not the human interface. The dynamic description shows which operations have to be performed by the user in order to invoke an instance of the service, to change the parameters of the service during the call, and to release the call.

The most commonly method used to represent this exchange of actions is that of an overall specification and description language (SDL) diagram. An SDL diagram represents, step-by-step, all actions, and all actions following from them (action-flow diagram). It is important to note that the dynamic description treats the network as a single entity, that is, no flows within the network are considered.

In Figure 2.2 it is shown where the dynamic service description is applied. The dynamic service description details the interactions between the parties involved in a call, not the human interface. In MAGIC it was felt that the dynamic service description should reflect more closely the concepts applied in the static service description, and therefore a more modular approach was used.

MAGIC provides individual SDL descriptions for each individual PT that may be involved in a service ([MAG03], pp. 160 - 296). By combining the local views, a global view can be obtained. Thus for each PT an individual SDL description is prepared, describing the finite state machine (FSM) for that PT's operations within the service. This leads to the model depicted in Figure 2.3.
In Figure 2.4 the relationships between the dynamic description flows and the flows obtained in stage 2 (which will be described in the next section) is shown. The concepts of the dynamic description operations and transactions is likely to facilitate the mapping of user requests into information flows at the network access.

**2.3. Stage 2: Identification of the Network Functional Elements and the Information Flows between Them**

In CCITT Recommendations I.300 ([CCI300]) stage 2 has been described in five steps:

1. Derivation of a functional model
2. Information flow diagrams
3. SDL diagrams for functional entities
4. Functional entities
5. Allocation of functional entities to physical locations
The objective of the stage 2 description is to identify the functional capabilities (entities) within the network and the information flows among them needed to support the services described in stage 1. The basis for the stage 2 description developed in MAGIC is the draft CCITT Q.73 Recommendation ([CCQ73]), in which a functional separation of call and bearer control (CC and BC, respectively) is introduced. This model has been modified to introduce a third functional level, called resource control (RC), between CC and BC. RC provides control over resources such as multicasting and conference bridging. Having these resources in a separate functional level allows simpler information flows within each level, and means that functional equivalence with CCITT Recommendations Q.27xx\(^5\) ([CCQ27]) can be achieved at BC level.

In Figure 2.5 the stage 2 functional model is depicted. Clearly visible are the three layers of control. OLEX, TLEX, BCTEX and RCTEX stand for different functional nodes in the service. They will be explained in chapter 3.

![The MAGIC functional model.](image)

The MAGIC project uses object oriented techniques as a means of modeling what has to be manipulated (controlled) to support the services described in stage 1. Having identified the object to be manipulated (controlled), the control system to manipulate them (including the signaling protocols) is easier to design.

The traditional concept of a call is as a single end-to-end object. the MAGIC project has produced a conceptual design, in which a call is a collection of call objects, along with the relationships between them. A global view (which can be helpful for understanding) and local views are developed. The local views are fundamental, since each control entity, except for the control entities at the OLEX, has only a local view of the service.

The MAGIC project has followed a slightly different approach in the stage two description as described here. All steps described here are encompassed in the MAGIC design. The results

\(^5\) [MAG03] mentioned the Q93.B Recommendations [CCQ93B]. This, however, has been changed into the ITU Q.27xx Recommendations.
of the stage 2 descriptions of the MAGIC specification methodology that are relevant to this thesis will become clear in chapter 3.

2.4. Stage 3: The Detailed Description of the Signaling Protocols

In stage 3 the information flow and the SDL diagrams from the stage 2 output form the basis for producing the signaling system protocol and switching recommendations. The steps in stage 3 will need to be repeated for each service where because of different allocations of functional entities to physical locations, different protocols and procedures are needed. MAGIC has provided detailed protocols for the user network interface (UNI).

2.5. Conclusions

The MAGIC project has followed the CCITT Recommendations I.130 ([CCII130]) for developing signaling protocols. In stage 1 the services to be offered by the B-ISDN network were described. The services range from simple (static) telephone point-to-point services to complex open services, including multiparty and multimedia. For the static service description a service description framework (SDF) is developed. In the SDF, the service attributes are classified in levels. The dynamic service description is described at the user application level, describing the interactions between the various party types. In stage 2 the network functional entities are defined, along with the information flows between them. In the MAGIC project a new functional level-resource control- is introduced between the existing call and bearer control. This allows for functional equivalence at BC level with the CCITT Recommendations Q.27xx ([CCQ27]). It also allows for simpler information flows per level. Stage 2 will be explained in more detail in the next chapter. Stage 3, the protocol designs, has been completed for the user to network interface.
3. The MAGIC Functional Model

The MAGIC functional model describes the three different levels of control (call, resource and bearer control), which take all actions necessary to set up, maintain and release a call. Call control coordinates and manages the negotiation of telecommunication services amongst a set of users. It instructs resource control to find and allocate the necessary resources for a call. Bearer control is, conform to CCITT I.130 Recommendations, introduced for the setup and release of connections on a link-to-link basis. Resource control is responsible for the control of specialized resources, such as multicastr (distributers) and bridges.

Two functions are needed in the functional model:
- the translation of the call control model to a model, which is understandable to resource control (implicit modeling, with no references to actual resources).
- the translation of this model to physical resources (explicit modeling, object oriented, with the objects being the physical resources).

Each entity involved in the service has only a local view of the call. The local view of resource control contains (explicit) parts the parts of the service request that the RC entity has translated and an implicit (or explicit) representation of the service request that is passed on by the RC entity to another RC entity. CC peer-to-peer, CC-RC and RC peer-to-peer communication is based on the Atomic Actions concepts of Commitment, Concurrence and Recovery.

Large parts of this chapter are copied from [MAG06].

3.1. Introduction

In stage 2 of the specification methodology, the functional model is introduced. The functional model consists of three layers of control: call, resource and bearer control. In this chapter the functionality of each of the control layers is specified.

In the global view modeling of a call, two functions become apparent: one that does the translation of the call control view of a call onto a model that is understandable to resource control, and one that translates this model into resources. Both translation functions and representations formats are described.

Each functional entity involved in the call has limited information on the call. The breakdown of the global view of the call into the these partial views is explained. The view of a call of resource control entities is explained in more detail in this chapter. The views of call and bearer control entities can be found in Appendices 1 and 2.

Large parts of this chapter are copied from [MAG06], where more information can be found.

3.2. Layout of the MAGIC Functional Model
The MAGIC functional model is depicted in Figure 3.1. In this model three levels of control are defined: call control (CC), resource control (RC) and bearer control (BC). MAGIC has introduced the resource control level, with which the CCITT Q.73 ([CCQ73]) descriptions for the call and bearer control can be applied with as little change as possible.

**Figure 3.1: The MAGIC functional model.**

### 3.2.1. Call Control

The essential function of call control is to coordinate and manage the negotiation of telecommunications services among a set of users. Call control deals with objects in the control plane (c-plane, see Figure 3.2) that do not have a direct relation to the physical objects in the user plane (u-plane, see Figure 3.2). Special resources, and bearer connections, are both part of the u-plane between user endpoints. Call control needs to map the objects of the c-plane onto a model which is understandable to resource control (using the adaptation function). Separation of call and resource control simplifies the definition of the call control part, since it can be defined independently of the requirements to control special resources located within the network. This results in greater commonality between the call control parts of the protocols at the UNI and NNI.

In the functional model, depicted in Figure 3.1, only the CC entity (and not the BC entity) from the agent is logically connected to the network. This is according to CCITT Recommendations Q.2931 ([CCQ2931]), which describes a monolithic protocol. There is no RC entity at the User Agent, as there are no specialized resources to be managed outside the network by network RC entities.

---

6 This model is equal to the model in Figure 2.5.
7 See Figure 3.1.
3.2.2. Resource Control

The main function of resource control is to translate the representation of the service request from call control onto objects of the u-plane. The objects controlled by resource control are referred to as special resources, to avoid confusion with the objects controlled by bearer control, which are referred to as normal or ordinary resources. In the remainder of this paper, resources stands for special resources (controlled by resource control), unless mentioned otherwise. By defining a separate RC entity in the network, function-specific protocols can be defined and network services and resources may be managed more efficiently. It becomes possible to allocate special resources at an RC node which is not involved in call control (RCTEX in Figure 3.1).

3.2.3. Bearer Control

Separation of resource control from bearer control facilitates backward compatibility with Release 1 signaling at the bearer control level. For backward compatibility with Q.2931 ([CCQ2931]), the bearer control part of Q.73 ([CCQ73]) is limited to controlling switching and transmission resources on a link-to-link basis. Reservation and allocation of special resources requires additional functions that are not provided by existing bearer control protocol definitions. It also implies that multicasting, a function which (in future) can be supported by simple nodes, will be controlled at resource control level.

3.3. Global Modeling Overview

The modeling overview of stage 2 of the MAGIC specification methodology is composed out of three major parts: one part describes the control relationships (with controlling and controlled objects), another the physical u-plane and the last one the local view graph. This is depicted in Figure 3.2.

---

8 Actually, this requires a much more thorough investigation than suggested in the MAGIC project. In [PLAT41M] this problem is investigated.

9 This model is a improved version of the global modeling overview presented in [MAG06], page 27. The model presented here makes the situation of the two translation functions more clear. The MAGIC model can be found in Appendix 10.
Two functions are needed. One is called the mapping or adaption function and is located in call control. It maps the call model objects to the resource control objects. The other function is situated in resource control. It maps these service description objects onto the control units of resource control (elements of the u-plane).

The adaptation function of call control translates the call control view of a call into an implicit representation of that call, which will be handed over to resource control. Resource control at its turn receives this representation which has to be translated into real controlling objects, or which will (partially) be transferred to another node which can translate at least part of the representation or which knows which node(s) do(es). The modeling of both representations is explained in the next section.

### 3.4. Explicit versus Implicit Model

Call control is responsible for determining which telecommunication service should be implemented, and how it is implemented. Resource control is responsible for the implementation. In this section the representation of the requests from call control to resource control and from resource control to implement a telecommunication configuration is discussed.
An implicit representation of the requested telecommunication configuration contains no references to the real implementation resources. For example, for a logical connection between two parties that use different codings a conversion is needed. Implicit representation of this configuration is possible by putting an attribute coding on the endpoints of the logical connections. The need for a converter can be derived (by the entity that controls the physical converter or an entity which knows an entity which does control the converter) by comparing the codings of the endpoints.

An explicit way of representing the required telecommunication configuration consists of a number of objects that represent special resources and connections between the ports of the special resources and the users. Explicitly requesting implementation of telecommunication services may require more knowledge of available resources than a functional entity (call control or resource control) has. Providing the capability to implicitly specify resources empowers the client to let the server make the decision about not only which types of resources to use, but also which specific resources are required. The RC entity that handles a request will have a better idea about the availability of its own resources and hence letting this entity specify the necessary resources may result in a more efficient use of resources (this will become clear in chapter 4).

Call control passes an implicit representation of the service request to resource control. Each RC entity strips of the part of the service request it can make explicit (i.e. translate to physical resources). The remaining part of the service request is sent to the next RC entity, until the total service request is made explicit. Each RC entity therefore only has a local view of the service request. The local view modeling is explained in the next section.

3.5. Local View Model

Each control entity has only a local view of the service. In this view a control entity only has an explicit view of the part of the service request it has translated, and an implicit view of the information it passed on to other controlling entities. An important exception to this rule is the controlling exchange (CEX), where the global view (implicit/explicit) of the service is kept (both at CC and RC level). All modifications of the call are transported to the controlling exchange. At this point CC passes the service request to resource control. The most likely place for the CEX is the OLEX, although this is not always the case. For easy description we will consider the OLEX to be the CEX throughout the remainder of this thesis.

The call control information flows are based on the CCITT Q.851 recommendations ([CCX851]) on Commitment, Concurrency and Recovery (CCR, see Appendix 3). For RC peer-to-peer communication the same Atomic Action mechanism as for call control is applied. The difference between both are the objects and operations contained in the Atomic Actions. The CC-RC connection is client/server based, thus the Atomic Action concept can be applied here. The RC Atomic Action tree can be seen as a subtree of the CC Atomic Action tree. In the first phase resource control has to set up the whole configuration, hoping

---

10 Which is according to the modeling: the total service request is composed by the CC entity at the CEX and received by the RC entity (from call control) at the CEX.

11 See rule in [MAG13], pag 142, lines 11-18.
call control will commit. If not, all connections and reserved resources have to be released, thus restoring to the state before the Atomic Action will be difficult.

The CC entity at the originating switch (OLEX, see Figure 3.1) has a global view of the total service. As mentioned in 3.3, it translates the model into a representation that is understandable to resource control and passes it on to resource control. The receiving RC entity then supplies the resources required it can supply locally, and passes the remainder of the service representation to other RC entities in the network. In this fashion the resources will be acquired in a distributed manner and each RC entity only has explicit information on the part of the service resources it supplied.

All entities at all three levels therefore only have a local view of the service request. The local view models of call control and bearer control can be found in Appendix 1 and Appendix 2, respectively. The local view of resource control is explained in the next section.

3.5.1. Resource Control Local View

The objects of the resource control local view are elements of the u-plane. To model the user plane model (UPM), an object oriented technique was used. Only the aspects which are important for the signaling are modeled. The objects of the UPM are obtained by invoking network resources. Resources are under the control of the resource management, and resource management is therefore responsible for allocation and de-allocation of resources. If the signaling system wants to create a UPM object, the object will automatically generate a request to the resource management entity.

As mentioned, call control performs a mapping of the call control local view model objects onto a service request which is understandable to resource control. Resource control in turn translates the service request to physical resources. The physical resources are the objects of the UPM. The list of objects can be found in [MAG06], pp. 76 - 79.

3.6. Conclusions

In this section the MAGIC stage 2 functional model is described. The model consists of three layers of functional entities. The first layer, call control, deals with the negotiations concerning the setup, maintenance and release of a call. Resource control, the second layer, deals with the allocation and de-allocation of the resources, necessary for the call. Bearer control provides the actual connections.

The global view modeling of a call is presented. In it, two functions are necessary:

- The translation of the elements of the call control view (control plane) of the call to a model, which is understandable to resource control. The modeling is implicit, with no direct references to resources.
- The translation of this model to physical resources, the explicit model. The explicit model is an object oriented model. The objects are part of the user plane.
Each controlling entity has a local view of the service only, except for the call and resource control entities at the OLEX, which have an overview of the total service request. The CC entity at the incoming exchange (OLEX) defines the service that is being demanded, and sets up the links to the other endpoints. The CC entity performs a mapping of the call model objects onto a more concrete RC request, which is passed to the RC entity. Each RC entity strips of the representation what it can supply locally, and sends the rest to another RC entity, which can supply (another part of) the service request, or knows which entity does. In this fashion each resource control only maintains an explicit view of the part of the service request it has translated itself and an implicit (or explicit) view of the part it passed on. The information is sent in implicit form, if the service representation has not been translated to actual physical resources, and in explicit form if it has.

Specifications of the information flows at and between CC and RC level are based on the Atomic Actions concepts of Commitment, Concurrence and Recovery. It provides a mechanism for distributed applications, such as signaling.
4. Criteria for Resource Control

Resource control must find resources necessary for the translation of a service request from call control. The time used to find the necessary resources, the costs of the resources and the costs of the connections made are the criteria that define the quality of a translation. Variables are defined to scale the different possible translations. As an optimal translation may not be possible (because the required resources are not available), thresholds are defined to limit the number of possible translations. Threshold values for resources used and total connection costs are network dependent and are therefore best estimated empirically. They are defined at resource control level. Their values may depend on the time threshold, which is defined at call control level. Rules for the calculation of thresholds in the progress of the configuration of the service request are given. Whether the actual connections costs are identical to the costs calculated at resource control level, depends on bearer control. The total connection costs, calculated at RC level, are therefore an estimate and may not represent the actual costs.

4.1. Introduction

Algorithms have to be designed, by which resource control can effectively translate a service request to physical resources. In this section, the efficiency obtainable with these algorithms is examined. The algorithms in fact represent the functionality of resource control as a whole. For the optimization of resource control functionality these algorithms should optimize the way resource control translates the service request to physical resources. In order to define optimization, criteria have to be defined to scale the quality of the configuration. Along with these quality parameters, threshold parameters can be defined to limit the acceptability of any configuration, considering the individual criteria. The definition of the optimization (quality) and threshold parameters is the topic of this section. These parameters will be used in chapter 8 to construct the algorithms.

4.2. Call Implementation Criteria

In [PLA411], section 5.4.3, it is mentioned that there are fundamentally two criteria for resource allocation:

The first is a global allocation problem. We can describe this problem as in [CHO95]: Given a network containing service-oriented resources and call description which requires some of these resources, allocate the physical resources, including the bearer connections necessary to support the communications specified in the call. As shown in [CHO95], this problem is NP complete and an optimal allocation algorithm to solve it is exponential. There is also an heuristic algorithm given by Chow in this article. However, there has not
been much research on this particular problem and there may be different approaches which will yield better results than Chow's.

In this problem, the cost of resource and the cost of communication necessary to connect resources and parties together must be taken into account. ... part of the solution to the global allocation problem includes the solution to the logical-to-physical mapping problem. However there is another aspect of the problem which remains, even if there is a 1:1 mapping of logical to physical devices, and even supposing that all devices are identical. The problem that remains is where to allocate the resources, which may have great influence on the costs for the call.

From this description two criteria for resource control directly emerge: the cost for the (special) resources used and the cost for the connections, necessary for the call. A third criterion can be added: time. With this criterion the length of the call setup can be delimited, to prevent the network from searching infinitely for an optimal configuration in terms of the two other criteria. The quality criteria to be used in resource control are therefore:

- **TIME**: Time is an important factor which can be pre-defined by the user.\(^\text{12}\) A user requesting a quick setup of a call can put more weight on the time it takes to allocate the resources needed for the call. This can result in higher charging, because a sub-optimal solution may be used when considering the two other criteria.

- **(SPECIAL) RESOURCES USED COSTS**: If different configurations in terms of resources are possible, it is important to investigate the difference in costs. An easy example is a data communication channel between two users, one with encoding type A, and one with encoding type B. One translation to resources is a single A-to-B converter, another translation might be a combination of A-to-C and C-to-B converters.

- **TOTAL CONNECTION COSTS**: Perhaps the most important criterion for resource control. When configuring a service request, the (special) resources have to be allocated in such a manner that a minimum cost for the connections is achieved. Total connection costs stands for the total product of length of the connections and bandwidth allocated for the specific service (or a derivative of the bandwidth). One could say that this is actually a functionality which should reside at bearer control, but by allocating resources (and thus deciding where the resources for a service are located), resource control is capable of influencing the connection costs in the service.

Note: Whether the actual connections costs are identical to the costs calculated at resource control level, depends on bearer control. It is possible that, because of e.g. malfunction of connections rerouting is applied. This is not known at RC level. The total connection costs, calculated at RC level, are therefore an estimate and may not represent the actual costs. Introducing this knowledge at resource control level would significantly complicate resource control and is therefore excluded.

A new term is introduced:

\(^{12}\)This is a possibility which is chosen for here. In MAGIC it is not mentioned whether this should be possible or not.
CONFIGURATION: The configuration is the translated representation of the service request from call control, in which all implicit parts of the service request have been translated to physical resources (see physical/explicit representation). Each configuration is unique for a call in terms of resources and locations of the resources.

4.3. Using the Criteria for Resource Control: Definition of the Variables

With the criteria defined in the previous section, some variables can be introduced to implement these criteria. These criteria can be used to construct algorithms resource control uses for the actual translation of a service request to physical resources.

The time criterion is a simple threshold parameter that defines the time available for the setup of the service request. Time will not be included in the algorithms as an explicit variable. Time will be used as a parameter in the choice between different algorithms (see chapter 8), used for the translation from the logical description of the service request into physical resources (representation: $T_i$). If algorithms, in which time is used as an explicit parameter, become available, the parameter $T_i$ can be applied as well.

Total connection costs and resources used costs variables can be used to define the quality of the configuration, speaking in terms of resources used and the locations of these resources (determining the connection costs). The variables playing a role in the way the configuration request is configured are:

- $B_t$: absolute threshold for total connection costs ($B$ for bandwidth)
- $b$: occupied bandwidth penalty constant
- $R_t$: absolute threshold for resources used costs
- $R(Converter)$: resource penalty for converter
- $R(Combiner\_2)$: resource penalty for 2-input combiner
- etc. (for each specialized resource).

4.4. Definition of the Thresholds

Still unsolved remains where and how the thresholds are defined. They can be defined either by the party that sets up the call, at call control level or at resource control level. Defining the thresholds at call control level, or by the party that sets up the call, would imply a change in the various interfaces, already defined (the UNI interface and the CC-RC interface), which is not preferable and should only be done if the advantages are larger.

The time threshold is best defined at the CC entity at the OLEX. This CC entity has a full overview of the service, and it is also the place where the decision is made to post, pre or simultaneous allocate the resources, necessary for the service (influencing the time needed to setup the call). It implies a small change in the CC-RC interface: the addition of a parameter, indicating the time threshold value. With this parameter RC entities are able to choose between different algorithms for the translation of a logical representation of a service request.
to physical resources. These algorithms will differ in time consumption (and in optimality obtainable in terms of resources used and total connection costs). How the time threshold value is determined is a topic of further research but not a part of the resource control functionality anymore. An additional advantage of defining the time threshold before invoking resource control is that this leaves the option open to let the party, that sets up the call, (indirectly) define the time threshold. This can be desirable if users of the network should be able to define (within certain boundaries) the time they allow for the setup of a service request (e.g. for emergency calls). Call control also maintains a certain time-out, inherent to the atomic action, which also introduces a time dependency (an absolute threshold).

The total connection costs and resources used costs thresholds could also be defined at call control level. Besides the necessary change in the CC-RC interface, this would require certain knowledge of the complexity of the service requested, speaking in terms of resources and connections. This in turn implies certain knowledge of resources at call control level, which is not the case in MAGIC (and not desirable). Therefore, both the total connection costs and resources used costs thresholds are defined at resource control level. The preferable place would be the RC entity at the OLEX, where the total view of the service request is present and where the service request is received from call control. However, it depends on the knowledge the RC entities have where the thresholds will be defined. This knowledge will be investigated in the next chapter (chapter 5). With the total view of the service request and some translation tables involving all parties, the distances between the parties, the ASM’s (see chapter 2) and the connections of the parties to the ASM’s, values for both the total connection costs and resources used costs thresholds can be defined. It is preferable to keep the algorithms for these calculations as simple as possible, to minimize the time taken to calculate the values for both thresholds.

A disadvantage of defining both thresholds at resource control level is that a party, involved in the call (in general any party, but most likely, the party that requests the call), has no influence on both the resources used and connection costs for a service, as they depend on the complexity of the service only. This can be disadvantageous if the user should be able to propose the preferred resources to be used (e.g. an AT&T user preferring the use of AT&T bridges instead of other bridges, which may be closer by in the network). However, a network is in general defined as the total hardware of a single operator only (see chapter 5) and thus this is not a problem which can be solved at resource control level, but which must be solved at call control level.

For the total connection costs threshold, different algorithms are possible:

1. For each ASM the total connection costs threshold value is defined as follows:

   \[ B_t = \beta_1 \sum_i \left( \sum_{ij} d_{ij} b_{ij} \right) \]

   \( d_{ij} \): distance between users \( i \) and \( j \), where \( i \) is a sending party and \( j \) is a party receiving the signal from party \( i \).

   \( b_{ij} \): bandwidth of the connection between users \( i \) and \( j \), where \( i \) is a sending party and \( j \) is a party receiving the signal from party \( i \).

   \( \beta_1 \): multiplication constant.

   The total connection costs threshold value for each ASM is equal to the sum of all distances between each sending party and all parties, receiving the signal from the
sending party, multiplied with a constant. The value of the multiplication constant is left open to the network operator.

Note: Estimating the costs for connections as a linear product of the length of the connections and the bandwidth of the connections is a simplification of reality. Aim is to obtain a quick estimate of the real connection costs for the definition of the threshold value. For this, the described algorithm suffices.

2. For each ASM and each sending party, construct a minimum spanning tree with all parties receiving the signal from the sending party. The threshold value is then defined as follows:

\[ B_t = \beta_2 \sum_i d(MST_i) b_i \]

- \( d(MST_i) \): total length of the minimum spanning tree for sending party \( i \).
- \( b_i \): The bandwidth for the tree belonging to sending party \( i \).
- \( \beta_2 \): multiplication constant.

In this algorithm the length of the connections is optimized. No attention is paid to the fact that not each party may need to receive the signal in full bandwidth, but the algorithm can be improved to include this as well.

The second algorithm is more complex, as it makes use of the minimum spanning tree algorithm. It may be nearer to the optimal configuration, but pays no attention to the possibility that necessary resources may not be available in the network defined by the minimum spanning tree, which might even deteriorate the result. As the freedom in both algorithms lies in the multiplication constants, which are network dependent and must be obtained empirically, no direct advantage of the second algorithm is apparent, and therefore the first algorithm is chosen.

The definition of the resources used threshold value is more complex, because it would require total resource control knowledge to be able to determine a practical value. Besides, the value would be estimated again with a multiplication value, which has to be obtained empirically, and therefore the algorithm should be kept as simple as possible. An easy algorithm for the resources used threshold for each ASM would be:

\[ R_t = \rho R(Bridge_{_N}) \]

- \( R(Bridge_{_N}) \): The price (in terms of resources) of a bridge with \( N \) inputs and outputs, with \( N \) being the number of parties involved in the abstract service.
- \( \rho \): Multiplication constant.

It is clear that the total connection and resources used costs threshold values are defined per ASM. It is therefore chosen to keep a separate total connection and resources used costs threshold values for each ASM. This implies that in the RC-RC interface for each ASM a separate total connection costs and resources used costs threshold value is passed on.

A last important note is introduced by the way the time threshold is defined. If a quick call setup is required, a quick algorithm is preferred. This however, implies that no guarantee can be given that the optimal configuration in terms of resources used costs and total connection costs is found. To emphasize on this, and to stress the importance of the time criterion in these setups, it might be desirable to enlarge the number of possible configurations by increasing both threshold values (this enlarges the number of possible configurations which are allowed). This requires dependability of both these values of the time parameter, that is
passed on. An easy way to implement this would be to make the $\beta_i, \rho$ parameters dependent of the $T_i$ value.

### 4.5. Thresholds through the Implementation Phase of a Call

As a call setup progresses, the service request arrives at different RC entities. The thresholds should be evaluated at all times, and changed if necessary. The following rules apply:

**Time threshold:** The time threshold is a parameter which is defined at call control and is not changed during a call. It is merely a parameter to choose between different algorithms (see chapter 8).

**Total connection costs threshold:** The threshold value is decreased with the connection costs for the part of the service request that is already implemented (made explicit). At a split of the service request to multiple RC entities, the threshold value for the service request for receiving RC entity $x$ for each ASM equal to:

$$ B_{tx} = B_t \frac{\sum_i \left( \sum_j d_{ijx} b_{ijx} \right)}{\sum_y \left( \sum_i \left( \sum_j d_{ijy} b_{ijy} \right) \right) } $$

$d_{ijx}$: distance between users $i$ and $j$, where $i$ is a sending party and $j$ is a party receiving the signal from party $i$, for all parties in the service request for RC entity $x$.

$b_{ijx}$: bandwidth of the connection between users $i$ and $j$, where $i$ is a sending party and $j$ is a party receiving the signal from party $i$, for all parties in the service request for RC entity $x$.

$B_t$: threshold value before the split.

$B_{tx}$: threshold value for RC entity $x$ after the split.

The relative part of the threshold value each receiving RC entity gets is equal to the costs of the connections of the service request it has to implement relative to the total connection costs needed by all receiving RC entities.

If the service requests to various RC entities (from a single RC entity) are sent in sequential order, the threshold values can be recalculated after each reply. The summation over $y$ is over all remaining RC entities that still have to implement a service request.

**Resources used costs threshold:** The threshold value is decreased with the values for the resources that are reserved. At a split, the threshold value for the service request for receiving RC entity $x$ equals:

$$ R_{tx} = R_t \frac{N_x}{\sum_y N_y} $$

$N_x$: number of parties in the service request for RC entity $x$.

$R_t$: threshold value before the split.

$R_{tx}$: threshold value for RC entity $x$ after the split.
The relative part each receiving RC entity gets is equal to the number of parties in the ASM relative to the total number of parties involved in the service requests to all receiving RC entities. If the service requests to various child RC entities are sent in sequential order, the value for $R_i$ can be recalculated at all times. The summation over $y$ is over all remaining RC entities that still have to implement a service request.

If, as of a result of a failed attempt to implement a part of a service request, reserved resources and connections have to be released, the total connection costs and resources used costs thresholds are increased with the values represented by the released reservations.

### 4.6. Conclusions

The criteria that define the degree of optimality of a service request configuration are the time used to implement the service request and the weighted sum of the resources used for the configuration and the total connection costs.

As an optimal configuration may not be possible, thresholds have to be defined to determine which solutions are acceptable. The time criterion is implemented by maintaining a time out at call control (inherent to the atomic action) and passing a parameter, with which resource control can choose between different algorithms, which (may) differ in time consumption. The parameter is defined at call control, which leaves the user free to influence the time allowable for the call setup. It requires a minor change in the CC-RC interface, as defined in the MAGIC project. Algorithms, which use time as an explicit parameter, can be used as well.

The total connection costs and resources used costs thresholds are both defined at resource control level, as they both require certain insight into the actual (possible) configuration of the service request. They both consist of a simple algorithm, including multiplication constants which have to be obtained empirically. The multiplication constant may be dependent on the time criterion parameter value.

Optimization can be achieved by checking possible configurations in decreasing order of quality, quantified by the criteria defined in this chapter. Threshold values can be used to mark the maximum allowable excess of the eventual configuration over the optimal configuration.

Algorithms are given to calculate threshold values in the case a service request splits to multiple service requests, and for the release of made reservations of resources and connections.

Whether the actual connections costs are identical to the costs calculated at resource control level, depends on bearer control. The total connection costs, calculated at RC level, are therefore an estimate and may not represent the actual costs.
5. RC Network Modeling

RC entities need information on the layout of the network and the locations of resources and users in the network to be able to translate the service request from call control to resources, using the various criteria. A (in this case distributed) form of resource control knowledge is necessary. In this chapter a network configuration, with a planar layout of subdomains (local domains) is presented. Each local domain consists of nodes and endnodes. The endnodes are interconnected to other local domains through interconnection trunks. In the network model a distributed platform solution for the distribution of resource control knowledge is presented. In this platform each RC entity has knowledge of the layout of the local domain it belongs to, and of the locations of the users and resources, present in the local domain. RC entities at endnodes have the ability to obtain information on the other parts of the network through a database. These RC entities are located at so-called endnodes.

5.1. Introduction

Resource control requires knowledge on the locations of resources and users and the layout of the network, in order to translate a service request. As the resource control is distributed throughout the network, a distributed implementation is necessary. In this chapter the distribution of resource control knowledge is investigated. It can be solved using distributed platforms. Several examples are given in Appendix 4. These alternatives are compared for various criteria, including their influence on the criteria for resource control, as mentioned in chapter 4. In this chapter one example is chosen as a reference model throughout the remainder of this thesis.

Before a model of distributed resource control knowledge can be constructed, a network model is necessary. In such a model, the network layout, the locations of users, the locations of RC entities and the relations between them are explained. The network model is called the domain topology throughout the remainder of this thesis.

5.2. Domain Topology

For the modeling of the distribution of RC knowledge a network model, the domain topology, must be defined. The topology of the domains investigated for resource control in this paper is depicted in Figure 5.1.

The domain consists of a collection of sub-domains, called local domains. All users and resources are located in these local domains. The local domains are interconnected via interconnection trunks. These interconnection trunks access the local domains at endnodes. Within a local domain, all RC entities have a direct peer-to-peer relationship with one another.
In the view of the RC entities at endnodes, each local domain, except the one it belongs to, is a concentration of nodes, with neglectible distances in between. Therefore, all parties belonging to a certain local domain, involved in a call, are, when viewed from outside of the specific local domain, considered so close together that their inter-distances are $\theta$. A prerequisite for this assumption is that the length of the interconnection trunks is in fact much larger than the length of the connections within a local domain.

**5.2.1. Terminology**

**DOMAIN:** A domain is a combination of local domains, interconnected by interconnection trunks. A domain is the total part of a network where the RC entities have a (direct) peer-to-peer relationship with one another, and thus have (the ability to obtain) information on the resources elsewhere in the domain. This is usually the total domain of a sole network operator (administration domain), although this is not a restriction.

**LOCAL DOMAIN:** A local domain is a part of a total domain (a domain), in which each RC entity has (direct access to) the total knowledge of the layout of this local domain, as well as the locations of all users and resources present in this domain. The local domain consists of endnodes and nodes.

**INTERCONNECTION TRUNKS:** The interconnection trunks are the trunks that interconnect the local domains in a domain. Interconnection trunks access local domains at endnodes.

**USER END(POINT )NODES:** User endnodes are switching nodes with users connected to them (LEX).
**SWITCHING ENDNODES:** The endnodes in the local domains connect the interconnection trunks to the other (switching) nodes in the local groups, and are thus actually gateways to the rest of the domain. It is very likely that there will be more than one endnode per local domain, because, if there is only one endnode and this node fails, communication on RC level with RC entities in other local domains has become impossible. It is important to note that all endnodes and nodes, with users connected to them (user endnodes), must support resource control.

**USER GROUP:** A user group is a group of endusers from an RC entity point of view. This view may be different for RC entities belonging to other local domains. For each RC entity, the users are grouped. All users belonging to other local domains are combined into one user group per local domain, for the local domain to which the RC entity in question belongs all users on the same user endpoint node are combined into one user group. This means that the user groups are identical for all RC entities belonging to one local domain. See Figure 5.2 for a visual explanation.

**LOCAL VIEW:** The local view is the knowledge each RC entity has of the resources and users in the local domain it belongs to, as well as the interconnection scheme of this local domain.

**LOCAL NODE:** In the view of a specific RC entity, a local node is a node belonging to the same local domain.

![Figure 5.2: Example of user groups.](image)

### 5.2.2. Validation of the Model

The advantage of this model is that it is a planar view, which is a future view of
networks, in which no hierarchy is present ([DO094]). It that it can be expanded in two ways:

- By addition of new local domains;
- By introducing new layers. The domains of layer 1 become the local domains of layer 2, etc. See Figure 5.3 for a visual explanation.

In the remainder of this thesis only the single layered structure is considered. Expansion to a model with more layers is achievable by redefining the endnodes and the local domains.

Figure 5.3: Layered expansion of the network.

5.3. Example Distributed Platform for Distributed Resource Control Knowledge

In Figure 5.4 an example distribution of resource control knowledge is depicted. In this example each RC entity has knowledge of the layout of the local domain it belongs to, as well as the locations of users and resources within the local domain. This knowledge is represented by the local view. Each RC entity at an endnode has knowledge of the interconnection layout of all local domains, that is, knowledge of all the local domains, their interconnections and the length of these connections. This is represented by the 'interconnection scheme array'. In this same array, the users are grouped per local domain. Additionally, each RC entity at an endnode has access to a global database. In this database the knowledge of the layout of the total domain is present. Also the locations of all users and resources available are stored.

For other solutions for distribution of resource control knowledge, refer to Appendix 4. In this Appendix five different distributions of RC knowledge are compared on various criteria. One criterion is the influence the modeling has on the criteria for the translation of a service request, as introduced in the previous chapter. Other solutions are envisaged as well. More
information can be found in distributed platform solutions ([BUK95], [CAR95], [YU95]). The network model does not delimit the number of possible designs.

Validation for the selection of the model presented here lies in the fact that the model introduces no restrictions on the translation of a service request by resource control (by optimizing the model) and the hard- and software demands for the implementation of such a model seem acceptable. The time, needed for the translation of a service request can be minimized by improving the hard- and software.

![Diagram](image)

**Figure 5.4: Implementation example - limited access to resource database.**

### 5.4. Conclusions

In this chapter a network model is presented. The network (domain) consists of a set of local domains, which are interconnected through interconnection trunks. Each local domain consists of a collection of nodes and endnodes. The interconnection trunks are connected to the endnodes. All users and resources are located in the local domains. Layered expansion of the model is possible.

In the presented distribution of resource control each RC entity within the domain has knowledge of the layout of the local domain it belongs to, as well as the locations of the users
and resources that are present within the same local domain. The RC entities at the endnodes have knowledge of the interconnection scheme of all local domains, as well as the locations of all users with respect to the local domains. Via access to a global database information on the availability of resources in other local domains can be obtained. The model introduces no restrictions on resource control in the translation of a service request, when considering the criteria, mentioned in the previous chapter, and the hard- and software demands for the implementation of the model seem acceptable.

The model for distributed resource control knowledge presented here is but an example. More examples, and criteria for comparison can be found in Appendix 4 or in reviews on distributed platforms. The actual design of the resource control knowledge is not delimited by the network model.
6. The Functional Model Interfaces

The functionality of resource control depends on the interaction with the environment. For a single RC entity, the environment consists of the CC and BC entities and peer RC entities. The communication over the CC-RC interface and between peer RC entities should include all necessary information for resource control to optimally translate a service request. The interface is configured in the simplest possible form, without letting call control make any decisions which might reflect upon the eventual configuration (in terms of resources used and connections made). Each abstract service is represented by a black box, with input/output relations for all involved users. User signal flows are considered 'finished' before they leave the network. From implementation point of view, a difference is made between user signals, which are dynamically composed, and static user signals. The interface is updated to include the time threshold parameter. The maximum number of parties is introduced as an optional parameter. With this parameter, precautions can be taken by resource control so that the signal flow to users is not interrupted during the addition or deletion of other users (only necessary for bridging and combining of signals). The implicit endpoints have been removed. Return messages have been optimized for their information contents.

All mentioned modifications to the CC-RC interface apply to the RC peer-to-peer interface as well. The physical representation (in which a translation to resources is made, but the resources are not located) is introduced for the RC peer-to-peer communication. The interface has been updated for the total connection costs and resources used threshold parameters. Implicit connections have become redundant between implicit elements.

The RC-BC interface is not finished. It should be extended to include the possibility for RC entities to order BC entities to give a list of all nodes in a connection and all BC entities involved in this connection. The interface will be corrected in the PLATINUM project.

6.1. Introduction

The functionality of the resource control (RC) entities depends on their interaction with their environment. The interaction environment is depicted in Figure 6.1. This model, the conceptual functional model, is equal to the stage 2 model depicted in chapter 3, with the input/output parts of the entities separated from the coordinating part. This separation makes it possible to identify the coordinating function on each level, as well as the input/output relations.
The resource control functionality lies within the RCc. Here the decisions concerning the configuration of a service request are made. For resource control, in order to be able to make these decisions, information on the service request is necessary. This information is passed from call control to resource control at the OLEX and between peer RC entities during the setup of the call. Also, bearer control needs to be informed to make the necessary connections. In this chapter the service primitives from CC to RC (CCc-RCc), the communications between RC peer entities (RCo-RCi) and the service primitives from RC to BC (RCc-BCc) are described. The syntax for the CC-RC and RC-RC interfaces can be found in Appendix 5. In Appendix 6 the ASN.1 description of the CC-RC and RC-RC interfaces are described.

### 6.2. General Remarks towards Modifications to the MAGIC Model

A few general assumptions are made here. These assumptions will lead to the decisions made in the remainder of this chapter.

1. Call control does not make any decision that is part of resource control functionality. This means that in the translation from the call control local view model to a model, which is understandable to resource control, all information, necessary to configure the service request -without any limitations induced by the translation- is passed on. It implies that the interface between call control and resource control becomes very simple and the adaption function as presented in [MAG06] becomes redundant.

2. A signal, carrying user plane information, delivered to a terminal is considered ‘finished’. This implies that different information flows from different parties to a party (in a single abstract service\(^{13}\)) are composed into one signal before it leaves the network. For video signals, for example, this implies that a user receiving video signals from different (other) users to construct a windowed picture does not need a terminal which can compose the

\(^{13}\) or synchronized abstract service group.
windowed picture from the separate signal flows. This windowing function is performed within the network.

3. The functionality of both a bridge and a combiner will in future surpass the functionality as envisaged in the MAGIC project. In the PLATINUM project a first attempt is made to describe an audio bridge. In future, bridges will be able to compose the signals to each of the outputs from all of the inputs at any time during the call. Parties involved in the call have signaling access to the resources to define the status of the signals. This kind of information involves modifications to the call configuration and is therefore control plane information (see section 1.1.2). With these bridges and combiners, the functionality of a network, as mentioned in the previous point becomes possible.

4. For modifications of calls it is desirable maintain the resources already allocated as much as possible, to minimize control information flows, modification times and reservation, allocation and release of resources. This may deteriorate the quality of a configuration (in terms of total connection cost and resources used cost), but it significantly simplifies the control over resources. In the MAGIC project no remark on this subject is made.

### 6.3. Entity Relations in One Stage Expressed in Numbers

In [MAG13] the entity relations in one stage are given for each stage. However, this model has the shortcoming that it does not deal with synchronization of abstract services. There is one RCc (Resource Control coordinator) per abstract service, so there can be no synchronization between different abstract services, because they are functionally separated at RC level. To overcome this problem, there is only one RCc per group of abstract services requiring synchronization.

In this section the entity relations in one stage are given for each functional stage.

#### 6.3.1. CCc Viewpoint

There is one CCi and one CCc per call, in a 3 level stage somewhere in the network.

There are \( i \) \((i \geq 1)\) CCo’s per CCc. \( i \) equals the number of addressed parties.

There are \( j \) RCc’s per CCc. \( j \) equals the number of synchronized Abstract Service group instances needed in the call.

#### 6.3.2. RCc Viewpoint

There is \( 0 \) or \( 1 \) RCi and \( 1 \) RCc per synchronized Abstract Service Group instance. The \( 0 \) RCI only applies in the originating exchange where the request comes in from the CCc.

There are \( k \) \((k \geq 1)\) RCo’s per RCc. \( k \) equals the number of addressed RC entities in the next stage.
There are $m$ BCc's per RCc. $m$ equals the number of connection points or connection endpoints needed in the abstract service group.

### 6.3.3. BCc Viewpoint

There is 1 BCc per connection point or connection endpoint. In the case of direction splitting (one bi-directional connection coming in, two -opposite- unidirectional connections going out), two BCc's are needed.

### 6.4. The CC-RC Interface

#### 6.4.1. Modifications to the MAGIC Model

In the MAGIC project, three different models for the CC-RC interface have been presented, one in [MAG06], one in [MAG13] and one in the 'Stage 2 Discussion Part' in [MAG13]. The model presented in [MAG13] is erroneous and incomplete, and is therefore not used. The model in [MAG06] is complete, but the mapping rules, presented in chapter 5.3.5 of [MAG06], restrict the resource control in its functionality\(^\text{14}\). This problem is solved with the transport model as presented in the third model (from the ‘Stage 2 Discussion Part’ of [MAG13]). However this model would imply two new translations in the MAGIC model: one translation of the CC view of the service to the transport model, and one translation of this transport model to the service request (placed to resource control). Still, this model does not comply with the first assumption mentioned in section 6.2 (call control makes no decision that is part of resource control functionality). This pre-assumption implies that the model will consist of a black box description of each ASM, with all parties and input and output relations for all connected parties. The elements used to describe this are endpoints and implicit bridges (and implicit synchronization groups for abstract service instances requiring synchronization). The implicit connections, distributers and combiners have become redundant in this modeling, making the model very simple. However, the model uses only elements of the RC peer-to-peer interface.

Another small change in the model is the omission of the implicit endpoints. Since the explicit information of the endpoints (that is: the exact locations of the endpoints) is known by CC and must be passed on to RC, it is of no use to introduce implicit endpoints. These explicit endpoints can be used in the (implicit) model of the service request, that is passed on from CC to RC.

A third modification involves the way the signals, the parties receive, are composed. In MAGIC, two kinds of ‘downstream access channels’ (see Appendix 1) were introduced: the composite DME’s and the component DME’s. A composite DME is related to an abstract service as a whole, whereas a component DME accesses information transmitted by an individual remote user or loops back information from the local user. Mappings from remote users are dependent on party service links (see Appendix 1). Component mapping gives the

---

\(^\text{14}\) See Appendix 7.
user more flexible control over the information coming from participating sources. It is more likely to be used in windowed signals.

With the difference between composit and component mapping, a distinction is made between dynamic and static signals from an implementation point of view. A dynamic signal is a signal where the user is free to compose the signal it receives from the abstract service from all parties involved in the abstract service, where it has party links to, at any time during the call. This in contrast to static signals, where the signal is static once it has been set up\(^\text{15}\). Component mapping is the obvious choice for dynamic signals, composit mapping is the choice for static signals.

Because this property (static/dynamic) has great influence on the way the service can be configured and the signal a user receives is considered 'finished' (see section 6.2), it is necessary to pass this status of the signal on to resource control. Therefore, for each implicit bridge and combiner (only for the RC-RC interface) the status of the signal (composit or component) will be passed on in the CC-RC interface (and RC-RC interface, respectively). In chapter 8 (and Appendix 9) the necessity of this extra information will become clear. Additionally, with component mapping, it may be possible that a party has component downstream mapping of a limited number of signals coming from parties involved in the abstract service. Therefore, in the case of component mapping, a list of the signals of the parties the receiving party has access to is mentioned (access list). This information is only necessary to correctly configure the service request. Once the translation to physical resources is made, this information is redundant. Permission rights of parties\(^\text{16}\) is a CC level problem.

Note: Combiners and Bridges will, in future, be able to supply the functionality for composit mapping, as depicted here. This functionality implies that the connection costs can be optimized by minimizing the bandwidth of the input signals. The combiner or bridge must somehow be informed of this. The option envisaged here is that this is implemented in the algorithm within the combiner or bridge.

Another modification in the CC-RC interface is the time parameter, called the time threshold value parameter (see chapter 4), with which resource control can be informed of the importance of the time criterion for the specific call setup. It is applicable per service as a whole.

Modified also are the contents of the reply messages (RETURN and COMMIT, respectively). In these messages only the necessary parameters are included. These are the parameters defining the atomic action and references to all implemented optional parts.

The final modification is introduced as a result of the following problem: For distribution of a signal the addition or deletion of a receiving party to/from the distribution does not result in the interruption of the signal flows to the other receiving parties (see [MOE96]), as distributors can be reserved and released without the interruption of the signal flows (these actions imply the addition or deletion of an address in a multicast). For combining or bridging of signals this is generally not possible, because the introduction or

\(^\text{15}\) Not considering the modifications of the call, where parties are added to or subtracted from the abstract service and the signal can be composed again.

\(^\text{16}\) Permission rights involving modifications on resource control level, such as the rights to modify the status of signal flows in the call.
release of combiners and bridges implies the rerouting of the signals flows (resources have to be released or introduced, whereas distributors can be modified). There are services however, where it is not possible (determined by the service aspects) that the signal flows are interrupted.

For these cases a solution is found by preconfiguring the call in such a manner that the addition of parties to the abstract services does not affect the already existing signal flows. Deletion of parties implies that resources can only be released if no user signal flow is left over. Depending on the service request, resource control has to take the following actions:

- In the case of bridging, a bridge with the number of connections equal to the maximum number of parties involved in the abstract service is allocated. The addition or deletion of a party results in the modification of the outputs to all receiving parties.
- For combining, an extra input needs to be left over as long as the maximum number of parties is not reached. Only the combining algorithms for the outputs of the combiners have to be modified in the case of the addition or deletion of a party from the service. A combiner may be released if no signal flow is left over.

The actions are only for component mapping, as for composit mapping the freedom of configuration of the service request is so large that each modification will (generally) results in an interruption of the signal flow (all connections have to be updated for the new bandwidth, all outputs have to be recalculated, etc.).

The modification involves the optional addition of a maximum number of parties value to the description of the implicit bridges and combiners in the interface. It is a call control problem to decide whether to introduce this value in the service request. Resource control does not create it by itself, as it has no insight into the concerning properties of the call.

6.4.2. Mapping Rules for the CC-RC Interface

With the remarks mentioned in section 6.4.1, the mapping rules have become obsolete. Each abstract service is represented by an implicit bridge. This decision resulted from a PLATINUM progress meeting. An alternative option would be to include implicit connections, combiners and distributors if from the service description it was absolutely clear that the service was a plain point-to-point, point-to-multipoint or multipoint-to-point connection. This derivation from the service description would very likely be faster than the translation at RC level, where no knowledge of the service description is present. However, the translation does involve resource control functionality and it also unnecessarily complicates the CC-RC interface. Therefore the first and simplest model is preferred over the second.

6.4.3. The Objects of the CC-RC Interface

The CC-RC interface contains the following objects, with the corresponding attributes:

**IMPLICIT PART**
• i_Bridge [1]
  Abstract Service id
  Reference to an incoming connection instance for each input
  Reference to an outgoing connection instance for each output
  For each output:
    Signal format (composit/ (component + accesslist + max number of parties)
    Mixing algorithm
    Time Threshold

• i_Sync group
  List of Abstract Services

EXPLICIT PART

• e_Endpoint
  Address of physical location
  For each port:
    -input/ output
    -format type

Notes:
1. More mixing algorithm attributes are required if the users request different mixing algorithms.

6.4.4. Additional CC-RC Interface Information Flow

For the UNI a monolithic interface has been designed (see chapter 3). This implies that the BC at the User Agent and the BC at the originating or terminating exchanges (OLEX and TLEX) can only communicate via the CC entities. This in turn implies that, for the CC-RC interfaces at these exchanges (OLEX and TLEX), information concerning BC must be exchanged. This information is copied directly from the MAGIC model presented in deliverable 13.

For error messages the same information structure, as presented in MAGIC, is used. Advantage of this structure is that it is network independent and thus left open in design to the operator.

A final piece of information are the so-called atomic action parameters, in which some parameters for communication on call control level is exchanged as well (necessary for synchronization etc.).

6.4.5. Syntax of the CC-RC Interface

The syntax of the CC-RC interface can be found in Appendix 5. In Appendix 6 the ASN.1 descriptions of the CC-RC interface are given.
6.5. Information Flow of the RC-RC Interface

6.5.1. Modifications to the MAGIC Model

As for the CC-RC Interface, the model introduced in [MAG06] is taken as starting point because the model introduced in [MAG13] is erroneous and incomplete. The modifications made on the CC-RC interface also reflect on the RC-RC interface: the implicit endpoints have become obsolete and for the implicit bridges and combiners the signal format (composit/component) for the receiving parties in an abstract service is transported in the information flow between peer RC entities. For further explanation refer to 6.4.1. The time threshold value is introduced as well, for resource control needs a parameter indicating the importance of the time criterion. The time threshold value is included in the parameter list of each service request. The modifications optimizing the contents of reply messages are introduced. Finally the option to include the maximum number of parties parameter, for implicit bridges and combiners, is copied from the CC-RC interface definitions.

The RC at the OLEX\textsuperscript{17} receives a service request from the CC, containing explicit endpoints and furthermore only implicit bridges (and implicit sync groups for synchronization). After translation, the RC entity decides which part of the service request can be made explicit, and tries to implement this part of the service request. After reservation of the necessary resources, the rest of the information is passed on to the next RC-node, until the full service request has been implemented.

For the information flows between peer RC entities a new term is introduced, next to the implicit and explicit representation:

**PHYSICAL REPRESENTATION**: In the physical representation the implicit model of the service request has been translated into physical resources. These resources in the representation have not been located yet and therefore do not represent actual resources within the network. This representation can be useful when only one implementation is possible\textsuperscript{18}, to delimit the number of translations made (a time-saving effort). The physical representation is identical to the explicit representation, with the exception that the actual address of the resources is not available. Therefore the physical representation can be transmitted in the current MAGIC interface, by using for example a dummy address for the resources. Large advantage is thus that no modifications to the MAGIC model are necessary\textsuperscript{19}.

The information passed on between peer RC entities is therefore in 3 possible formats: implicit, physical or explicit, where the physical format is conceptually the same as the explicit format. The receiving RC entity determines the format by checking the addresses of the elements (a dummy address implies the element is in physical format). For explicit connections no physical variant is possible on RC level. A connection can first be made

\textsuperscript{17} Remember this is actually the CEX.
\textsuperscript{18} Depending on the algorithm chosen.
\textsuperscript{19} An additional advantage, not present in the modeling, is that RC entities can make decisions on the configuration of calls in the network domains of other operators, without any knowledge of the locations of resources in the specified domain.
explicit when both the elements it connects are explicit. If at least one of the connected elements is physical, the connection is physical and cannot be configured yet.

Note: The physical representation of a connection cannot bear threshold information. Therefore, an RC entity is able to translate to physical connections only if it has insight into the length of the connections. E.g. in the model of the domain, the distances within a local domain are more or less negligible compared to the length of interconnections. Therefore, an RC entity having information on the length of interconnection trunks can make decisions concerning translations to physical connections, without knowing the actual distances within the local domains, when searching for resources there. This implies that an RC entity receiving a service request with physical connections cannot pass this request on if it fails to make these connections.

Another change is that an implicit connection is no longer used between two implicit objects or between an implicit object and an explicit object. This is done because all information of an implicit connection can be transported in another implicit object. All implicit connections, connected to another implicit object are therefore redundant. This implies however, that all port references of the implicit bridges, combiners and distributors must refer to the endpoints (explicit endpoints or implicit objects) they are connected to. These two alternations imply a change in the instance connections between the implicit objects (of the resource control local view), as given in MAGIC deliverable 6, section 5.3.3.1.3, page 88. The corrected version of figure S5.54 is given in Appendix 8.

An additional change is necessary in the RC information flow: total connections costs and resources used costs thresholds, as introduced in chapter 4. As these two thresholds are defined per abstract service, they must be passed on for each implicit object representing a (part of a) abstract service: the implicit connections, bridges, combiners and distributors. For normalized and specialized resources translated to explicit and physical representation no resources used costs threshold values have to be included, as the translation to (physical) resources has already been made and the cost (in terms of resources used or connections) is known.

If bearer control is able to reply with the actual length of the connections, it would be able to include the total connection cost threshold for the physical and explicit connections, as there is still uncertainty in the costs of the actual connections. This would require additional information flow in the bearer control plane. If it is desirable to dynamically control traffic at RC level, it is better to do this before resource control translates a logical model of the service into explicit resources. This could be accomplished by a dynamic distance penalty database for the connections between RC nodes. This option is therefore not included.

6.5.2. The Objects of the RC-RC Interface

---

20 Chapter 5: The lengths of connections within a local domain are negligible compared to the lengths of interconnection trunks.

21 Which has consequenations for algorithms for the search of the optimal location of a bridge (as will become apparent in chapter 8), because the distances can no longer represent geographical distances.
The RC-RC interface contains the same objects as the CC-RC interface, with the addition of the implicit distributors, combiners and connections, explicit connections, as well as the explicit or physical bridges, combiners and distributers, converters and routing groups. The objects, and their corresponding attributes are:

**IMPLICIT PART**

- **i_Connection [1]**
  - Abstract Service id
  - References to ports
  - Total Connection Costs Threshold
  - Resources Used Costs Threshold
  - Time Threshold
- **i_Combiner**
  - Abstract Service id
  - Reference to an incoming connection instance for each input
  - Reference to an outgoing connection instance for the output
  - Signal format (composit/ (component + access list + max number of parties)
  - Mixing algorithm
  - Time Threshold
  - Total Connection Costs Threshold
  - Resources Used Costs Threshold
- **i_Distributer**
  - Abstract Service id
  - Reference to an incoming connection instance for the input
  - Reference to an outgoing connection instance for each output
  - Time Threshold
  - Total Connection Costs Threshold
  - Resources Used Costs Threshold
- **i_Bridge [2]**
  - Abstract Service id
  - Reference to an incoming connection instance for each input
  - Reference to an outgoing connection instance for each output
  - For each output:
    - Signal format (composit/ component + access list + max number of parties
    - Mixing algorithm
  - Time Threshold
  - Total Connection Costs Threshold
  - Resources Used Costs Threshold
- **i_Sync group**
  - List of Abstract Services

**PHYSICAL/ EXPLICIT PART**

- **e_Endpoint**
  - Address of location (dummy address for physical representation)
For each port:
- input/output
- format type

• e_Connection [1]
  Format type
  References to ports
  Directionality

• e_Combiner
  Network address of location (dummy address for physical representation)
  Format type
  Number of inputs
  Reference to an incoming connection instance for each input
  Reference to an outgoing connection instance for the output
  Mixing algorithm for output

• e_Distributor
  Network address of location (dummy address for physical representation)
  Format type
  Number of outputs
  Reference to an incoming connection instance for the input
  Reference to an outgoing connection instance for each output

• e_Bridge [2], [3]
  Network address of location (dummy address for physical representation)
  Format type
  Number of inputs
  Number of outputs
  Format type
  Reference to an incoming connection instance for each input
  Reference to an outgoing connection instance for each output
  Mixing algorithm for each output

• e_Converter
  Network address location (dummy address for physical representation)
  Format type corresponding to input
  Format type corresponding to output
  Reference to incoming connection for the input
  Reference to outgoing connection for the output

• e_Routing group [4]
  Reference to connections

Notes:

1. Implicit connections are always unidirectional, explicit connections can be bi-directional. This is a arbitrary decision copied from MAGIC. Motivation is that bearer control can handle bi-directional links, whereas the implicit connections are unidirectional to allow for separate connections in both directions (e.g., if necessary converters are not allocated at the same node). If the instructions for the same connections are passed on to bearer control at the same node, only one BC entity is necessary.

2. More mixing algorithm attributes are required if the users request different mixing algorithms.
3. The physical and explicit bridges do not keep track of the permission rights of the parties to the signals of other parties in the same abstract service. Once the translation to the necessary resources is made, permission rights will be handled at call control level.

4. The e_Routing group is needed for the implementation of the i_Sync group.

5. For the explicit combiners, distributors and connections the format type can be optimized in the case the bandwidth is minimized (for composite mapping). For bridges and combiners the format type corresponds to the format type (in terms of bandwidth) of the output port.

### 6.5.3. Additional Information

As for the CC-RC interface, some call and bearer control information may exchanged as well.

Another addition is the provision of the necessary information flows for placing queries, as described in Appendix 4. The querying is based on a simple action: the RC entity placing the query, asks the receiving RC entity (belonging to another local domain) for the availability of physical resources. The receiving RC entity replies positively or negatively. The necessary control flows are shaded in the syntax, as they are only part of the total RC peer-to-peer control flows in implementation alternative 2 and only need to be implemented for RC entities at endnodes. The induced action sequence is as follows:

![Figure 6.2: Action sequence for the queries between RC entities.](image)

In the syntax the possibility for placing a query per object is introduced. This can easily be extended to include more than one object per query.

### 6.5.4. Syntax of the RC-RC Interface

The syntax of the RC-RC interface can be found in Appendix 5. In Appendix 6 the ASN.1 descriptions of the RC-RC interface are given.

### 6.6. The RC-BC Interface
After the RCc has made a part of the service request explicit (i.e. translated it into physical resources), it has to instruct bearer control at the specific node to make the necessary (explicit) connections. In this chapter the RC-BC interface is described. The model is copied directly from the descriptions in [MAG13].

A modification to the model is the inclusion of the option for RC to interrogate BC for the list of nodes in a connection, and the BC entity references for these nodes. This is necessary for the addition of a distributor. For further clarification refer to chapter 7. The model is not finished yet and in [PLA411] many questions were raised. In PLATINUM WP 4 a workforce has been assigned to clarify and correct this interface model.

6.7. Conclusions

Resource control may not be limited in the translation of the service request by the translation from the call control view model to a model, understandable to resource control. The representation of the service request, passed over the CC-RC interface is therefore in the simplest possible format: each abstract service is represented by a black box, with input/output relations for all involved users. User signal flows are considered 'finished' before they leave the network. From implementation point of view, a difference is made between user signals, which are dynamically composed (component DME) and static user signals (composit DME). The interface is updated to include the time threshold parameter. The maximum number of parties is introduced as an optional parameter. With this parameter, precautions can be taken by resource control so that the signal flow to users is not interrupted during the addition or deletion of other users (only necessary for bridging and combining and only for component signals). The implicit endpoints have been removed. Return messages have been optimized for their information contents.

All here above mentioned modifications to the CC-RC interface apply to the RC-RC interface as well. The physical representation (in which a translation to resources is made, but the resources are not located) is introduced for the RC-RC communication. The interface has been updated for the total connection costs and resources used threshold parameters. Implicit connections have become redundant between implicit elements.

The RC-BC interface is not finished. It should be extended to include the possibility for RC entities to order BC entities to give a list of all nodes in a connection and all BC entities involved in the connection. The RC-BC interface definitions will be corrected and completed in the PLATINUM project.
7. Action Flow Diagrams for Resource Control

Depending on the representation of the received service request and the knowledge of RC entities, their action flow diagrams may differ. Three different AFD's are described in this chapter:

- The RC entity at the OLEX receives the service request from call control. It calculates the resources used costs threshold and the total connection costs threshold for the part of the service request that belongs to the local domain, where this RC entity belongs to (local part of the service request). It implements the local part of the service request.

- The RC entities at the endnodes consult the database if necessary and send the service request parts to the concerning local domains. The RC entities at the originating endnodes receive the service request directly from the RC entity at the OLEX. They calculate the total connection costs thresholds for the remainder of the service request. If an RC entity at an originating endnode fails to set up (part of) the service request, the total service request fails.

- Other RC entities only receive explicit service requests and are therefore not involved in any decision making.

For modifications of calls where resources are released or reserved, several assumptions and rules are given. With these assumptions it is possible to maintain the distributed translation of a service request. Releases of calls are identical to the mechanisms, as described for setups of calls. Only modifications are in the contents of the messages.

RC-RC communications is always according to the principles of the two-phase AA mechanism, as presented in Appendix 3.

7.1. Introduction

Action flow diagrams describe, step-by-step, each operation an RC entity, involved in a call setup, has to perform. In detail, each action and decision is described, for each state the functional entity is in. Depending on the presentation of a service request (implicit, physical or explicit), which an RC entity receives, and the knowledge of the RC entity the action flow diagrams for RC entities may differ. In this chapter the action flow diagrams for RC entities are described. The flow charts are given in a description, according to the specification and description language (SDL) ([MAG06]). The descriptions are similar to the descriptions in [MAG06], pp. 64-72.

7.2. Terminology

GLOBAL DATABASE: The global database is the database with information on the total domain. It contains information on both the layout and the locations of all resources and users in the domain. No exact addresses of the resources are necessary in the database (see [MOE96]).
LOCAL DATABASES: The local databases are the databases where the information on the local domain is maintained (see Appendix 4): the User Endpoint Array, the Resource Array and the Local Nodes Array. Note that the Endnodes Interconnection Scheme Array (see Appendix 4) is not considered a local database. This to make a distinction between the databases at the non-endnodes and the endnodes.

CONSULTING A GLOBAL DATABASE: This term is introduced to describe the action in which the database, with knowledge on the total domain, is consulted. Remember this is dependent on how the resource control knowledge distribution is modeled (see chapter 5).

LOCAL CALL: A local call is a call where only parties from the same local domain are involved, and the call is set up in the same local domain (no third party call setup, with the third party belonging to another local domain than the parties actually involved in the call).

NON-LOCAL CALL: A non-local call is a call where parties from different local domains are involved, or third party call setups, where at least one of the parties involved in the call does not belong to the local domain where the party, that sets up the call, belongs to.

ORIGINATING ENDNODE: The originating endnode is the first endnode to receive (a part of) the service request. This endnode is therefore always located in the local domain where the service request is passed on from call control to resource control. There can be more than one originating endnode per call.

GLOBAL PART OF THE SERVICE REQUEST: The global part of the service request is the part that is implemented by the RC entity at the endnodes. Here the translation to physical resources is made, by consulting the database for the availability of the necessary resources and then sending service requests (in physical representation) to the RC entities at the endnodes of the local domains where the resources have to be allocated. For the parts of a service request that belong to the global part the following rules apply:

- For point-to-point connections ([MAG03]), there is no global part of the service request.
- For point-to-multipoint and multipoint-to-point connections ([MAG03]) all parties that belong to the same local domain are grouped into one group. The remaining configuration is the global part of the service request.
- For multipoint-to-multipoint connections ([MAG03]) the whole service request is global, unless at least half the parties involved in the bridging belong to the same local domain (in which case the service request is a local part for that specific local domain).

LOCAL PART OF THE SERVICE REQUEST: The local part of the service request is the part that of the service request that is optimally implemented (see chapter 8) at the local domain where the RC entity in question belongs to. In general all parts of the service request that do not belong to the global part are split into different local parts of the service request, one per local domain. The RC entities at the endnodes that configure a part of the service request, send these service requests in implicit representation to the RC entities at the endnodes of the local domains where these resources are to be allocated.

Note: There is one exceptional case in which the split of a service request into a global and local parts might deteriorate the eventual configuration. This case can be found in [MOE96].
7.3. General remarks

7.3.1. Remarks on RC Entities at Endnodes

The messages transferred from the RC entity at an endnode to the RC entities at other endnodes are always in implicit or physical representation. This allows the receiving RC entity to locally check for alternative solutions for the service request. It also implies that the RC entity at an endnode, when consulting a global database or placing a query, does not need addresses of the resources in the reply. This in turn implies that no addresses of resources have to be stored in the global database. If an RC entity consults the global database, the consult is always in physical representation. Therefore the global database requires no intelligence to be able to translate an implicit service request to actual resources, and only has to check whether the required resources are available within the boundaries set by the threshold levels. If the global database was able to translate implicit service requests, RC functionality would be located in the global database, centralizing resource control.

If the RC entity at an originating endnode fails to set up (part of) the service request, the total service request fails. This implies that all decisions concerning alternative possible implementations for the global part of the service request are made by the RC entities at the originating endnodes. It is however possible that a non-originating endnode must implement parts of a service request that cannot and should not be implemented in the local domain it belongs to. It must then be able to decide where to implement it (e.g. by consulting the database) and be able to send the message. An RC entity must be able to decide whether or not it should consult the global database, update thresholds and implement parts of the service request in other local domains. The following rules apply:

1. An RC entity receiving a service request in physical representation can only check for alternatives in the same local domain.
2. An RC entity receiving service requests in implicit representation for a local part of the service request can look for alternatives in other local domains, as long as the solutions are within the threshold boundaries.
3. An RC entity receiving service requests in implicit representation for a global part of the service request can look for alternatives in other local domains via consults to the global database, as long as the threshold values are not violated. In general, these cases imply concurrent setup of a global part of the service request with another, local part of the service request. One of these parts is best implemented in the local domain in question.
4. An RC entity receiving service requests in implicit representation for a local part of the service request of another local domain along with global parts that need to be configured, can send the service request to the concerning domain, as soon as all preceding parts are fulfilled (addresses are needed).

---

22 For the physical part of the service request only alternative locations of the resources within the same local domain can be evaluated, for the implicit part different configurations can be evaluated as well.
23 Note, however, that in the local databases the addresses of the resources are kept.
5. If a physical part of a service request fails in setup, the creator of the message must look for alternatives.

6. If an implicit part of a service request fails in setup at the creator, the service request fails, unless it is an implicit global part that is sent along with a physical global part. It is possible to totally reconfigure the global part of the service request.

### 7.3.2. Remarks on Thresholds

As mentioned in chapter 4, the total connection costs threshold value can be defined when the total overview of the service is known. For the RC entity at the OLEX, the total connection costs threshold can be defined for the local part of the service request only. The RC entity at the OLEX defines the resources used cost threshold and the total connection costs threshold value for the local part of the service request. The RC entity at the originating endnode defines total connection costs thresholds for the remaining part of the service request. This value should be large enough to include eventual local parts in other local domains, so that after these two computations of the total connection costs threshold value, no new computations are necessary.

Note: Each threshold value is an absolute boundary in terms of acceptability of a configuration and should only in rare cases prevent a service from being implemented.

An important aspect concerning threshold values is the way they evolve during the RC process. If some parts of the service request have been allocated, the values should decrease accordingly. First of all, the RC entity that makes a part of the service request explicit or physical, deducts the total connection costs and resources used costs threshold for these parts. If an RC entity sends a message in implicit format to other RC entities, the threshold values are calculated according to algorithms, as suggested in chapter 4. The RC entity that receives an implicit service request, does not give a reply containing information on the actual value of the criteria used (with which the originating RC entity can reuse the left-over threshold parts) because this would give the originating RC entity insight in the way a service request is configured. This is not wanted, as the service request was transferred in implicit format, implying that the originating RC entity has no insight into the configuration at all. It would increase the information the sending RC entity has of the call.

Therefore, RC entities receiving positive replies upon implicit or service requests deduct the threshold values passed in the (implicit) service request from the remaining threshold values, supposing the total threshold values included in the service request are used. When a part of a setup fails and all previously reserved resources and connections are released, the threshold values are increased to their original values.

### 7.3.3. Remarks on the Actions Sequences

Inherent to the atomic action model, all RC entity actions are valid as long as the time taken does not surpass a certain threshold level. If the time-threshold is surpassed, the CC at the OLEX notices this and stops the setup (ROLLBACK).
An RC entity in a call never delegates the mastership to another RC entity. This means that only the two-phase AA mechanism (BEGIN - READY - COMMIT) is valid for resource control communication (for further explanation refer to Appendix 3). Each RC entity is therefore parent from each RC entity it addresses. This results from the fact that the CC-RC interface is a two-phase AA mechanism, and a mixture of the two different AA mechanisms is only possible from a one-phase to a two-phase mechanism (otherwise RC entities would have multiple parent entities, see Appendix 3). It is also a logical consequence from the fact that each RC entity only passes that part of its service request that it cannot translate to resources itself to another RC entity. In this way the receiving RC entity can never have a broader view of the service request that the originating RC entity. The RC entity at the OLEX, which has a global view of the service request (see section 3.5), remains master of the call.

If the RC entity at the OLEX fails in configuring the local part of the service request (or part of it), this part becomes part of the global part of the service request, as the RC entity at the originating endnode must find an implementation for it (resources are located in another local domain than the parties involved and thus the RC entity at the originating OLEX is the only one which can perform this task). After locating the necessary resources, by consulting the database, this part of the service request is sent to the local domain(s), where the resources necessary are located.

7.4. Description of the Action Flow Diagrams

The RC entity at the OLEX receives the service request from call control. Upon reception, it calculates the total connection costs (only for the local part of the service request) and resources used costs threshold values and checks the local part of the service request. It allocates as much as possible and sends the remaining part of the service request to the RC entity at (an) endnode(s).

Each RC entity at an endnode which receives the service request directly from the RC entity at the OLEX is called an RC entity at an originating endnode. It calculates the total connection costs threshold value for the remainder of the service request, checks the global part of the service request and consults the database if necessary, until a solution for the total global part of the service request is available. It then sends the service request parts to the RC entities at all concerning local domains, reserving resources locally if necessary. When the whole service request is finished the RC entity at the originating endnode informs the RC entity at the OLEX.

The RC entity at an endnode which receives a service request from the RC entity at the originating endnode, tries to implement the local parts of the service request locally. If this succeeds, it informs the RC entity at the originating endnode. If it fails it tries elsewhere, if possible within the thresholds. Other parts of the service request it receives may also need to be implemented elsewhere. The functionality is therefore the same as for the RC entity at the originating endnode, which results in identical flow diagrams.

24 The service request it received from a peer RC entity or from call control (at the OLEX).
25 This implies that the RC entity at the specific local domain receives a service request involving parties outside its local domain.
All RC entities which are neither at the OLEX or at endnodes receiving a service request from the RC entity at endnodes belonging to other local domains, or from the RC entity at the OLEX, only receive explicit service requests from other RC entities.

The three distinguishable groups of RC entities differ in their functionality. This could be seen as a disadvantage, because three different action flow diagrams (AFD) have to be designed. The large advantage however is that all RC entities which cannot be at the OLEX or at an endnode in a call (thus all RC entities belonging to nodes which are neither endnodes nor have users connected to them) will proof to be very simple in design and therefore also in implementation. Large parts of the various algorithms can be re-used in other algorithms, making the design-phase less complicated. For each of the three kinds of RC functionality the algorithm is developed and the AFD is designed in this section.

It is possible that the OLEX is the originating endnode in the same call. This would imply an additional AFD for an RC entity combining the functionality of the RC entity at the OLEX and the originating endnode. However, this is can be implemented with the before described three groups of RC entities. If the OLEX is the originating endnode at the same time, the functional RC entity representing the RC entity at the OLEX invokes a new RC entity, within the same node, representing the RC entity at the originating endnode.

In this section the AFD's for the three distinguishable groups of RC entities are described, each in a separate subsection. Each subsection starts with a short description of the action sequence. Next the AFD is given, along with a detailed description of each step in the AFD. The atomic action tree, belonging to the AFD is given as well. It can be helpful in investigating the structure of the AFD's.

The AFD's are divided into several parts, separated as shaded boxes. These shaded boxes are re-used in the AFD's of other RC entities. They describe the common parts of the AFD's.

7.4.1. RC Entity at Non-OLEX and Non-Endnode

The AFD described here is for RC entities which do not meet the following requirements:

- The RC entity is at an endnode receiving an implicit or physical service request.
- The RC entity is situated at the OLEX.

In these cases, the RC entities only receive explicit service request and are not involved in the translation of a service request. Their only function is to reserve resources as ordered. In short, the action sequence is as follows:

A. Upon reception of the service request (BEGIN), reserve the resources, if necessary. If no resources are necessary, instruct bearer control to make the necessary connections and go to C.

B. If this fails, send a REFUSE back to the parent RC entity, else instruct bearer control to make the necessary connections.

---

26 An RC entity at an endnode may receive explicit service requests from another RC entity within the same local domain. In this case only resources have to be reserved.

27 This extra step is only necessary to allow the re-use of parts of this AFD in the description of other AFD's.
C. If this fails free the reserved resources and send back a REFUSE to the parent RC, else send a READY.

D. Upon reception of a ROLLBACK command from the parent RC entity free all resources and connections reserved, upon reception of a COMMIT allocate the reserved resources and throughconnect the reserved connections.

This leads to the following AFD:

![AFD Diagram](image)

**Figure 7.1: AFD for RC entity at non-OLEX and non-endnode.**

1. **BEGIN**
   - Receive explicit service request from parent RC entity

2. **resources?**
   - Check if resources need to be allocated
Instruct RM management to reserve the resources in the service request (arbitrary whether parallel or sequential)

Negative reply from RM that the desired resources could not be reserved

Positive reply from RM that the desired resources are reserved

Instruct BC to make the necessary connections (arbitrary whether parallel or sequential)

Negative reply from BC that the desired connections could not be reserved

Positive reply from BC that the desired connections are reserved

Instruct RM to free the reserved resources

Refuse the explicit service request from the parent RC entity

Inform the parent RC entity that the service request setup is ready

The parent RC entity instructs the RC entity to commit the service request reservations

The parent RC entity instructs the RC entity to rollback the service request reservations

Instruct RM to allocate the reserved resources

Instruct RM to free the reserved resources

Instruct BC to throughconnect the reserved connections

Instruct BC to release the reserved connections
7.4.2. RC Entity at the OLEX

The RC entity at the OLEX receives the service request from call control. Therefore the service request is always in implicit format. In short, the action sequence is as follows:

A. The first step is to define the total connection costs (only for the local part of the service request) and resources used costs threshold values.
B. The next step is to translate the local part of the service request to the physical representation (supposedly possible within the thresholds).
C. Now check the local databases for the availability of the resources needed in the physical representation.
D. Supposing the needed resources are available within the thresholds, send service requests in explicit format to the controlling RC entities.
E. When all the resources of the local part of the service request have been allocated successfully (and the thresholds have been updated), the RC entity calculates the threshold values for all receiving RC entities (at the endnodes) and sends the remainder of the service request to those receiving RC entities.
F. The RC entity at each of the originating endnodes reports back whether the remaining parts of the service request have been reserved successfully, and the threshold values are updated. If one (originating) endnode fails to set up its part of the service request the total service request fails and the RC entity at the OLEX frees all reserved resources and connections and sends back a REFUSE to call control. In the other case a READY is given to call control.

This results in the following AFD:
Receive implicit service request from call control

Define the total connection costs (for the local part only) and resources used costs thresholds
Translate the local part of the service request to the physical representation (using the algorithms)

Check if the necessary resources in the physical representation of the service request are available locally

Check if resources have to be allocated by the RC entity at the current node (the OLEX)

Update the local part of the service request that has been translated to explicit resources

Recalculate the total connection costs and resources used costs thresholds

Check if the local part of the service request is ready

Construct and send an explicit service request to an RC entity (in the same local domain)

Negative reply from the RC entity to the explicit service request

Reply from the RC entity that the required resources were reserved and the connections made
Check if the total service request is ready

Inform call control that the service request is ready

The CC entity instructs the RC entity to commit the service request reservations

The CC entity instructs the RC entity to rollback the service request reservations

Instruct RM to allocate the reserved resources

Instruct RM to free the reserved resources

Instruct BC to throughconnect the reserved connections

Instruct BC to release the reserved connections

Instruct child RC entities to commit the service request reservations

Instruct child RC entities to rollback the service request reservations

Construct the service requests for the receiving RC entity (part of total service request)

Calculate the thresholds for the child RC entity to be reached

Send service request (in implicit/physical representation) to the RC entity at the (originating) endnode

Positive reply from RC entity (at the endnode) that service request has been fulfilled
Update the thresholds

Negative reply from RC entity (at endnode) that service request could not be fulfilled

Free all reserved resources

Release all reserved connections

Send rollback message to all child RC entities

Negative reply to call control that service request could not be fulfilled

Note: A difference is visible between 40. and 25., both places where a message is sent to another RC entity. At 25. the steps compromising the construction and sending of a service request are combined into a single step, whereas 40. is preceded by 38. and 39., where the message contents (including the thresholds) are defined. At 25. an explicit service request is sent, and the receiving RC entity does not have to translate a part of the service request. At 40. the receiving RC entity receives a service request which still requires translation (implicit and physical parts). It has to create a new view, which is a further translation of the (implicit/physical) message it receives. The difference is made to make a clear distinction between the two cases.

In the AFD several shaded boxes, with a letter, are drawn. These are to identify points where the action sequence re-enters the AFD at another point, bearing the identical letter. An example is at the point where the shaded box with the letter A is introduced.

The AFD is not finished yet. The part at B, where an unsuccessful attempt was made to reserve selected resources, has to be filled in, as well as the part at C, where the resources required for the local part of the service request were not available.

From the point named B a decision must be made how to continue. Using the algorithms for the translation, alternative translations can be evaluated before sending the message to another RC entity. For C the steps are the same. The AFD now becomes:
Figure 7.4: AFD for RC entity at OLEX.

Check if other local configurations are possible for the remaining part of the local part of the service request.
7.4.3. RC Entity at Endnode

The two previous AFD's clearly demonstrate that the AFD of an RC entity receiving a service request, which still requires more translation, is considerably more complex in the design. The RC entity at the OLEX receives the service request directly from call control, thus the service request contains no direct references to resources. If the service request is explicit (as in Figure 7.1), the entire service request has been translated and no decisions have to be made. As a result the AFD for the RC entity at the OLEX is much more complex.

The final AFD is for RC entities at endnodes. Not only do they receive implicit service requests, they also have to decide where to send all the parts of the service request. Their knowledge is much wider, which results in an even more complex AFD. In short, the action sequence is as follows:

A. Calculate the total connection costs threshold, if necessary (only for the RC entities at the originating endnodes).
B. Upon reception of the service request, translate the implicit representation of the global part of the service request into a physical representation.
C. If necessary, consult the database to obtain information on the availability of resources.
D. If the reply from the database is positive (i.e. the configuration of the global part is possible), try to implement it by sending the service requests to the RC entities in question and reserving locally, if required. If not, go to E.
E. If all parts of the service request have been set up, send a READY to the RC entity at the OLEX. Upon failure, check if there are other valid configurations for the remaining global parts of the service request. If this has result, go to C.
F. Else release all reserved resources and free all connections and check if there is an alternative configuration for the total service request. If so, go to C.
G. Else send a REFUSE to the RC entity at the originating endnode.

This leads to the following AFD:
Figure 7.6: AFD for RC entity at endnode.

Free all reserved resources
Distributed Resource Control for B-ISDN Release 2/3

Release all reserved connections

Send rollback message to all child RC entities

Calculate the service requests for each of the receiving RC entities.

Update threshold values for all releases

Check if other configurations are possible for the total service request

Refuse the service request from the parent RC entity

Inform the parent RC entity that the service request is ready

The parent RC entity instructs the RC entity to commit the service request reservations

The parent RC entity instructs the RC entity to rollback the service request reservations

Instruct RM to allocate the reserved resources

Instruct RM to free the reserved resources

Instruct BC to throughconnect the reserved connections

Instruct BC to release the reserved connections

Instruct child RC entities to commit the service request reservations
Instruct child RC entities to rollback the service request reservations

Receive implicit/physical service request from parent RC entity

Define thresholds

Translate local parts of the service request

Check if a database consult is necessary

Consult the databases

Check whether the configuration is possible

Check if there are resources in the service request that have to be allocated locally

Check if other implementations are possible for the global part of the service request
7.5. Special Remarks on Modifications

For call modifications, the situation is considerably more complex. In the MAGIC model, as well as in the model presented in this paper, an RC entity receiving a service request, will configure part of it (or nothing at all), and send the remainder of the service request to another RC entity. In this manner, each RC entity 'shells' of that part of the service request it can configure, and knows which RC entity configured the remainder of the service request (because it was the creator of this RC entity by sending the remainder of the service request to it). In the case of a modification of a call, there are situations where a part of the call has to be reconfigured in the 'middle of the' configuration, that is by an RC entity which is both subordinate and superior of other RC entities within the call (see Appendix 3). For modifications a few assumptions are made:

1. For modifications of a distribution, the user plane (see section 1.1.2) signal flow to all parties (in the call), not involved in the modification, may not be interrupted, during the modification. Validation for this assumption is that this is generally not acceptable for the service (e.g. TV-distribution) and that it is easy to implement (see also chapter 6).
2. For the modification of a call involving component bridging or combining, the signal flow to all receiving parties may not be interrupted if notified in the call setup (i.e. maximum number of parties is given, see chapter 6).
3. For modifications involving the addition or deletion of a single party in any service, no more than one resource (excluding converters) may be modified -allocated, released or exchanged- (see section 6.2, item 4).
4. If an RC entity notices that the modification of a call requires the invocation of a new RC entity between itself and its child entity, it creates the new RC entity, sends the service request, including information on the former child entity. This former child RC entity now becomes child of the newly invoked RC entity.
5. If an RC entity notices that the modification of a call reduces its configuration part of the service request to a simple point-to-point connection, it kills itself and informs its parent and (possible) child RC entities of the new situation. After the modification is completed, only a point-to-point connection is to be controlled at the specified node and therefore no RC entity is necessary anymore (no special resources to be controlled). In cases where the RC entity in question has no child RC entities, the parent RC entity is able to deduct this from the remaining service request (no implicit parts left over).
6. If, for a distribution, a new distributor is needed, an RC entity along the path with minimal connection costs to the new party, instructs the concerning BC entity to send a list of all the nodes it passes along this connection. The RC entity picks the node with minimal connection costs to the new party. It invokes a new RC entity there. The newly invoked RC entity can contact the already existing BC entity at the node (in control of the connection) using the call reference and connection endpoint reference of the already existing connection.

7. Because of the above rules (4 and 5), RC entities must be able to interchange RC entity references. As these are not part of the syntax for the RC these are not included in the syntax notations for RC-RC communication (Appendices 5 and 6).

8. Information concerning updates on the new situation to RC entities, already in the call and involved in the new situation, are passed on after a COMMIT, to minimize traffic in the case of a ROLLBACK.

If the signal flows to users may not be interrupted in the case of combining or bridging of user signal flows, the action sequences are simple and can be found in [MOE96]. The schematic action sequences for addition or deletion of users in other cases, in which a resource is allocated or released, are given in Figure 7.8 through Figure 7.16.

Figure 7.8: Addition of a new combiner for local calls.
1. Order RC entity to release local party, connection B and combiner

3. Order RC entity to delete connection C and make connection A

**Figure 7.9: Deletion of a combiner for local calls.**

1. Investigate BC connection list for optimal location of distributor

2. Order RC entity to make connection to new party and allocate (upgrade) distributor and inform on connections A and B

4. Inform RC entity of new endpoint of connection A (instead of A + B)

**Figure 7.10: Addition of a new distributor for local calls.**
1. Inform RC entity on new endpoint connection A + B (instead of A)

2. Release local sending party and upgrade distributor

3. Order RC entity to make connection C and delete connection A, and new master (implicitly clear because message was sent by new master)

Figure 7.11: Deletion of a distributor for local calls.

Figure 7.12: Addition of a new combiner for non-local calls.
Distributed Resource Control for B-ISDN Release 2/3

1. Order RC entity to release local party

4. Upon commit inform RC entity on new master

---

Figure 7.13: Deletion of a combiner for non-local calls.

---

1. Check list of nodes from BC for optimal location of new distribution

2. Order RC entity on child entity and to make new connection

3. Allocate new distributor and make connection B.

4. On commit inform RC entity on new connection A instead of (A + B) and new master (implicitly)

---

Figure 7.14: Addition of a new distributor for non-local calls.
For situations where a resource is set free and a new resource is reserved, the situation is more or less the same. For distributors, a resource cannot be set free for a new one (see rule 1), unless it is a 2-distributor, in which case the resource is deleted and not replaced (remember this is the downgrading of a 2-distributor to a 1-distributor, which is a simple point-to-point connection). For combiners that are deleted and replaced by new combiners, the situation is more or less the same as in Figure 7.12, except the newly invoked RC entity has to make connections to all new RC entities:

**Figure 7.15: Deletion of a distributor for non-local calls.**

**Figure 7.16: Allocation of a new combiner for non-local calls.**
Problems like the one in Figure 7.16 become much simpler if the RC entity invoking a new RC entity has no child entities itself. In this case, the newly invoked RC entity does not have to inform child entities to make new connections, which it now can make itself.

Open problem remains how the exact information flow between RC and BC is in the case a new distributor is required and RC wishes to interrogate BC on a list of nodes in this connection. For more information on this, please refer to [PLA41M].

7.6. Special Remarks on Releases

Releases of calls are according to the mechanisms, as described for setups of calls. Only modifications are in the contents of the messages. The two-phase Atomic Action mechanism is still applied to allow ROLLBACKs.

7.7. Conclusions

In this section the action flow diagrams are described for the RC entities. The complexity of the AFD depends on the freedom in the translation of the service request and the knowledge of the RC entity. Freedom in the translation of the service request stands for the possibilities, the receiving RC entity has in translating the service request. An implicit service request has no direct references to resources, thus more possibilities are at hand in the translation than for an explicit service request, where only one alternative can be evaluated. Three different AFD’s are described:

- The RC entity at the OLEX, which receives the service request from call control and translates the local part (which should be implemented in the local domain to which the RC entity belongs). It calculates the threshold values for the resources used costs and the total connection costs for the local part of the service request.
- The RC entities at endnodes, which make decisions concerning the global parts of the service request through consults to the database (if necessary). The RC entities at originating endnodes receive the service requests from the RC entity at the OLEX. They define the total connection costs threshold value for the remaining part of the service request. They consults the database for the availability of resources for the global parts of the service request and send service requests to RC entities at endnodes of the corresponding local domains. Consults to the database are always in physical format (in terms of resources, but without actual addresses). Service requests between RC entities at endnodes are in implicit format (for local parts of the service request) or physical format (for global parts of the service request). If an RC entity at an originating endnode fails to set up part of the service request, the total service request fails.
- Other RC entities, which only receive explicit service requests and thus only reserve resources as ordered by other RC entities.

RC-RC communication is always according to the two-phase CCR atomic action principles (see Appendix 3).
For modifications on calls where resources are released or reserved, several assumptions and rules are given. With these assumptions it is possible to maintain the distributed translation of a service request.

Releases of calls are according to the mechanisms, as described for setups of calls. Only modifications are in the contents of the messages.
8. Using the Criteria in the Algorithms for Resource Control

In the framework, presented in chapters 5, 6 and 7, resource control uses various algorithms to translate the service request into physical resources. These algorithms implement the criteria, defined in chapter 4. The time criterion is used as a variable to choose between different algorithms. As the time criterion becomes more strict, a faster algorithm is chosen. This will in general deteriorate the configuration (in terms of resources used and total connection costs). In Appendix 9 two types of algorithms are given:

- Algorithms which optimize the configuration in terms of resources used and total connection costs.
- Heuristic algorithms which use considerable less time, but in general do not find the optimal configuration.

Algorithms, which use all three criteria as explicit parameters of algorithms, can be used in the framework as well.

8.1. Introduction

In chapter 4 the criteria were mentioned that define the quality of the translation from a service request to specialized resources (the configuration of the service request). This translation is performed by resource control. The three criteria are:

- The time used for the configuration of the service request;
- The total costs for the connections made in the configuration;
- The total costs for the (specialized) resources used in the configuration.

These criteria will be used in the algorithms, which resource control uses in the translation of a service request into a configuration. These algorithms can be embedded in the framework presented in the previous chapters. In this section an explanation is given how the various criteria are implemented in the algorithms. In Appendix 9 various algorithms are given.

8.2. Introduction of the Algorithms: Using the Criteria for Optimizing Resource Control

Depending on the knowledge of an RC entity and the format of the received service request (implicit, physical or explicit), the RC entity may have to translate (part of) the service request to resources (see chapter 7). This is the point where the criteria for resource control become important. By translating the service request, the criteria can be applied to define the quality of a service translation.
To optimize all three criteria within a single algorithm would yield a very complex algorithm. In [CHO95] it is proved that optimizing a translation when considering only the connection costs and resources used costs (and not time), yields an \textit{NP}-hard problem. The algorithm, OROAP, described in [CHO95] has three drawbacks:

- Not all possible configurations are considered in the algorithm. Actually, a few (not mentioned) restrictions lead to the supposed situation depicted in the article.
- The algorithm does not consider bridges.
- The time criterion is not explicitly mentioned. Time is optimized by trying to find the fastest algorithm.

Besides not explicitly using the time it takes to configure the service request as a criterion for the quality of the algorithm, the algorithm does not consider the full complexity of the problem in terms of possible translations to resources and connections. For many service requests, however, it is fairly easy to describe the full complexity of the problem and to deduct an algorithm which does find the optimal configuration, in terms of resources used and connection costs. In appendix 9, algorithms for the optimal configuration of service requests are derived for point-to-point, point-to-multipoint and multipoint-to-point connections. Time is optimized by applying the fastest algorithm (the algorithm which consumes minimum time in the progress of finding the optimal configuration). It is important to note that in those algorithms the time criterion is optimized only after the other two criteria are optimized (no concurrent optimization). For multipoint-to-multipoint connections with component downstream mapping, the possible number of configurations is relatively limited and can be described.

A second approach in describing algorithms is introduced. In this approach no attempt is made to describe the full complexity of the problem. Therefore, no guarantee can be given that the optimal configuration will be found. However, as the number of alternatives decreases, the time it takes to find the optimal configuration (of the alternatives taken into account) also decreases. In Appendix 9 various heuristic algorithms are introduced (for the same collection of service requests as mentioned above), which may yield poorer results in terms of connection costs and resources used costs, but require much less time.

The method of introducing all three criteria in the algorithms has become clear: depending on the time criterion a choice is made between a collection of algorithms. As the time restriction becomes less strict, an algorithm overviewing more alternatives may be used, to increase the quality of the configuration.

The complex service descriptions which do not belong to any of the before mentioned descriptions but which, in future, may become possible, remain unsolved. These can be summarized under the term multipoint-to-multipoint connections with composit downstream mapping\textsuperscript{28}. Here the algorithm presented in [CHO95], with the notified restrictions, is applicable. In Appendix 9 some guidelines are presented to help understand the full complexity of the problem and which may lead to some alternative algorithms.

\textsuperscript{28}The multipoint-to-multipoint connections with both component and composit downstream mapping can be handled accordingly. Each party with component downstream mapping must have access to the full bandwidth of the signals of the parties it has access to (from the access list (see chapter 6)).
8.3. Conclusions

The three criteria define the quality of the translation of the service request into physical resources. They are therefore in the algorithms resource control uses for this translation. The time criterion is used as a parameter, with which the choice between various algorithms is made. As the time criterion becomes less strict, the algorithm chosen will overview more alternative configurations, improving the quality of the translation in terms of resources used and connection costs. In Appendix 9 algorithms are given, optimizing the connection costs and resources used costs. In the same Appendix some heuristic algorithms are given, which overview less alternative configurations and may thus lead to poorer results in terms of connection costs and resources used costs, but improve the time used for translating the service request.
9. Conclusions

9.1. Conclusions

In this thesis, resource control, as introduced in the RACE project R2044 ('MAGIC') is enhanced and completed. Resource control is responsible for the translation of a service request (received from call control) to physical resources. Three criteria play a role in the translation of a service request to resources: the time it takes to translate the service request, the costs for the resources used and the costs for the connections made. A framework for this translation has been developed. In this framework resource control can apply algorithms, which perform the actual translation of the service request to physical resources.

The framework consists of the following elements:

- A network model: In this network model, the layout of a network is given. The network model in this thesis is a collection of sub-networks (local domains), which consist of nodes and endnodes. All users and resources are located in the local domains. The local domains are interconnected through interconnection trunks between the endnodes.

- The modeling of distributed resource control knowledge: The modeling of the RC entities knowledge can be solved using distributed platforms. In this thesis a model is presented as a reference model. In this model, each RC entity has knowledge of layout of the local domain it belongs to, and the locations of users and resources in that local domain. The RC entities at the endnodes have the ability to obtain knowledge on other parts of the domain through a database. The model of the network, as well as the model of the distributed RC knowledge, can be expanded into multi-level networks.

- The interface definitions: The interface definitions concerning resource control have been worked out in detail. At the CC-RC interface the service request is passed on as a black box description of the service, with input/output relations for all parties involved in the service. At the RC-RC interface three formats of communication are possible: implicit (no translation to resources has been made), physical (a translation to resources has been made, but the receiving RC entity is free to locate the resources) and explicit (translation to resources has been made, with the exact locations (addresses) of the resources). The format of the communication depends on the knowledge of both RC entities. The interfaces have been updated to include the thresholds for the various criteria. For combination or bridging of signal flows between parties, precautions can be taken to prevent interruption of the user signal flows during addition or deletion of parties. The maximum number of parties involved in the service value has been introduced in the interface definitions to support this function. From implementation point of view a distinction is made between the static or dynamic composition of signal flows to users during a call. This distinction is included in the service request at setup time. Implicit endpoints have become redundant and implicit connections are only used between explicit
Distributed Resource Control for B-ISDN Release 2

objects. The interface between resource and bearer control will be described in the PLATINUM project.

• Action flow diagrams (AFD's): Depending on the overview of the network and the format of the received service request, the action flow diagrams for different functional RC entities can be described. (1) The RC entity at the OLEX receives the service request from call control. It calculates the resources used threshold and the total connection costs threshold for the local part of the service request and configures the local part of the service request (for the local domain this RC entity belongs to). (2) The RC entities at the originating endnodes (receiving the service request directly from the RC entity at the OLEX) calculate the total connection threshold value for the remaining part of the service request. The RC entities at the endnodes decide where each part of the remaining service request should be implemented and can consult the database for the availability of resources in other local domains (if necessary). Information flows to and from the database are always in physical format, implying the database does not have to translate a service request to resources (which is a resource control functionality) and does not have to supply addresses. (3) All other RC entities only receive explicit service requests, making their action flow diagrams very simple. For modifications of service requests, special precautions have to be taken apply the enhanced modeling, as introduced in the MAGIC project. For addition of parties to a distribution, RC can consult BC for a list of nodes in a connection, for the optimal location of a new multicast (distribution).

The elements mentioned above form a complete framework for resource control. The framework is a basis, in which the resource control entities can apply algorithms for the translation of a service request to physical resources. The algorithms for the translation of the service request evaluate alternative solutions with respect to the three criteria mentioned above. A threshold value for each of the criteria is introduced as well. The threshold values indicate the range of acceptable configurations when considering the various criteria.

The time criterion is used as a parameter for choosing between multiple algorithms. Its value is defined at call control level. As the time criterion becomes more strict, the algorithms allow less alternative configurations (in terms of resources used and connection costs) to be evaluated. As the time available decreases, the threshold values for the resources used and total connection costs may increase, to allow a quicker setup.

The strength of the developed framework lies in the fact that it does not limit resource control in the search for an optimal translation of a service request: all criteria can be applied in algorithms. Optimizing all elements of the framework (e.g. minimizing the access time to the database) would result in optimal translations of service requests. Whether the actual connections costs are identical to the costs calculated at resource control level, depends on bearer control. The total connection costs, calculated at RC level, are therefore an estimate and may not represent the actual costs.

The threshold values and the variables for the criteria are free to be defined by the network operator. In this fashion, the network operator can dynamically scale the criteria. As better algorithms become available, they can be implemented without being limited by the framework. The framework is open to algorithms with the time parameter as an explicit parameter.
The network model used for the modeling of distributed resource control knowledge is very open: it can be expanded both horizontally (by adding more sub-domains) and vertically (by introducing the multi-layered concept). The actual modeling of distributed resource control knowledge is left open. The network model is only used as a reference model, without limiting the freedom of configuration of the network by the network operator. No limitations are induced by the modeling: the sub-domains and domains are not limited in size or number.

An additional advantage of the resource control knowledge and network modeling is that it allows for networks of different operators to be able to operate together, without requiring knowledge on each other's network (the network of each operator is in this case equal to a local domain in the modeling). RC peer-to-peer communication is thus possible between RC entities belonging to networks of different operators.

9.2. Modifications to the MAGIC Model

In this section the modifications to the MAGIC model, introduced in this thesis, are listed, along with the chapter number in which the modification can be found:

General modifications:

- The introduction of the criteria (and thresholds for these criteria) that define the quality of a translation of a service request by resource control: time used for the configuration of a service request and the costs of the configuration (resources used and total connection costs). [chapter 4]
- The introduction of the network model: the network (domain) is a collection of local domains (sub-networks). Each local domain is a collection of nodes and endnodes. All users and resources are located in the local domains. The local domains are interconnected via interconnection trunks at the endnodes. [chapter 5]
- The definition of a (reference) model for distributed resource control, in which each RC entity has knowledge of the local domain it belongs to and RC entities at endnodes have the ability to obtain information on other local domains via a database. [chapter 5]
- The split of a service request in a global and local parts. [chapter 7]
- Three different action flow diagrams for resource control that are a direct result of the modeling of the network: one for the RC entity receiving the service request from call control (at the OLEX), one for the RC entity at endnodes and one for the other RC entities. [chapter 7]
- One RCc entity per abstract service group instance instead of one RCc entity per abstract service instance. [chapter 6]
- The assumption that a user needs not modify or compose received signals. [chapter 6]
- The choice to modify at most one resource (excluding converters) in modifications, where one user is introduced or deleted. [chapter 7]
- The CC-RC communication, and the RC peer-to-peer communication is always according to the two-way atomic action principle. This implies that the mastership of a call is never delegated. [chapter 7]

Modifications to the CC-RC interface: [all chapter 6]
- The introduction of the time criterion threshold value.
- The choice between component and composit downstream mapping for signal flows to users, involving more than one sending party.
- The access list of parties and the option to include the maximum number of parties value in the definitions for component downstream mapping.
- The presentation of each abstract service (in the service request from call control to resource control) as a black box model, making the mapping rules redundant.
- No implicit endpoints.
- Optimization of the return (REPLY, COMMIT) messages.

Modifications to the RC-RC interface: [all chapter 6]

- The introduction of the time criterion threshold value for each implicit service request.
- The resources used and total connections costs threshold values for each implicit element representing an abstract service.
- The choice between component and composit downstream mapping for signal flows to users, involving more than one sending party (only for bridging and combining of signals).
- The access list of parties and the option to include the maximum number of parties value in the definitions for component downstream mapping.
- The introduction of the physical service request, in which the service request is translated to resources, but the resources have not been allocated.
- No implicit endpoints.
- No implicit connection between implicit objects or a combination of an implicit and explicit object (see Appendix 8).
- Optimization of the return (REPLY, COMMIT) messages.
- The option to transmit RC entity references.
- The option to place queries on the availability of resources in other local domains.

Modifications in the action flow diagrams: [all chapter 7]

- Rules for looking for alternative implementations.
- Rules for the threshold calculations.
- The reservation of the two-phase atomic action principle for all RC-RC communication.
- The split of a message into local and global part of the service request.
- The additional rules for modifications of service requests.

9.3. Recommendations

The model is finished and ready to be implemented. For the implementation, several recommendations can be made:

- When implementing resource control, keep the algorithms independent of the framework. In this manner, the resource control model can be improved if better algorithms become available and is not delimited by the algorithms currently at hand. Besides, it leaves the option open to let the network operator define the algorithms.
• Leave the definition of the thresholds and other variables, concerning the criteria, open to the network operator. This because the thresholds are network dependent (dynamic!) and with the freedom to define the algorithms and the variables concerning the criteria the operator has full control over the criteria for resource control and thus over quality of the translation of a service request by resource control.

• Keep the network model open to expansion. Introduce the option to expand the model both horizontally and vertically. Introduce the option to combine the networks of multiple operators.

• Keep the implementation of the distributed resource control knowledge independent of the implementation of both the framework and the network model. This leaves the network operator free to construct the resource control knowledge, being limited only by the layout of the network.
References


[CCI121] CCITT Recommendation I.121 'Broadband aspects of ISDN'.

[CCI130] CCITT Recommendation I.130 'Method for the characterization of telecommunication services supported by ISDN and network capabilities of ISDN'.

[CCI210] CCITT Recommendation I.210 'Principles of telecommunication services supported by ISDN and the means to describe them'.


[CCI327] CCITT Recommendation I.327 'BISDN Functional Architecture Aspects'.

[CCQ27] CCITT Recommendations Q.2761 'B-ISDN. Functional Description of the B-ISUP of SS.7'
CCITT Recommendations Q.2762 'B-ISDN. General functions of messages and signals of the B-ISUP of SS.7'
CCITT Recommendations Q.2763 'B-ISDN. SS.7 B-ISUP Formats and Codes'
CCITT Recommendations Q.2761 'B-ISDN. SS.7 B-ISUP Basic Call Procedures'
[CCQ73] CCITT Recommendation Q.73 'Basic call handling with separation of call control and connection control - Functional capabilities and information flows'.

[CCQ93B] CCITT Recommendation Q.93B 'Broadband ISDN access signaling protocol'.


[CCX851] CCITT Recommendation X.852 'Commitment, Concurrency and Recovery (CCR)'.


List of Acronyms

AA  Atomic Action/ Actual Association
AAL  ATM Adaption Layer
ACE  Access Connection Element
AFD  Action Flow Diagram
AS  Abstract Service
ASE  Application Service Element
ASG  Abstract Service Group
ASM  Abstract Service Module
ASN  Abstract Syntax Notation
ATM  Asynchronous Transfer Mode
B-  Broadband
BC  Bearer Control
BCA  Bearer Control Agent
BCc  Bearer Control Coord
BCTEX  Bearer Control Transit EXchange
CAO  Call Associated Object
CC  Call Control
CCA  Call Control Agent
CCc  Call Control Coord
CCR  Commitment, Concurrency and Recovery
CCITT  International Telegraph and Telephone Consultative Committee
CEX  Controlling EXchange
ChAO  Channel Associated Object
CRCG  Common Route Connections Group
DME  Downstream Mapping Element
ECO  Elementary Call Objects
FSM  Finite State Machine
ISDN  Integrated Services Digital Network
ITU (-T)  International Telecommunication Union (- Telecommunication Standardization Sector)
LEX  Local EXchange
MAGIC  Multiservice Applications Governing Integrated Control
NNI  Network Node Interface
OLEX  Originating Local EXchange
PE  Party Edge
PLATINUM  PLATFORM providing Integrated services to New Users of Multimedia
(formerly now as Project-X)
PRM  Protocol Reference Model
PT  Party Type
RACE  Research and Development in Advanced Communication Technologies in s Europe
RC  Resource Control
RCA  Resource Control Agent
RCc  Resource Control Coordinator
RCTEX  Resource Control Transit EXchange
RM  Resource Management
SDF  Service Description Framework
SDL  Specification and Description Language
SM  Service Module
TAO  Telecommunication Associated Object
TCS  Telecommunication Service
TE  Terminal
TEX  Transit EXchange
UME  Upstream Mapping Element
UNI  User Network Interface
USM  User Service Module
VC  Virtual Channel
VCC  Virtual Channel Connection
VCI  Virtual Channel Identifier
VP  Virtual Path
VPC  Virtual Path Connection
VPI  Virtual Path Identifier
WP  WorkPackage
List of Words

ACTUAL ASSOCIATION (AA): The AA describes the relation between the ASM's and the PT's. This link was added to the framework in a later stage. It is necessary that there is a direct link between the ASM's and PT's. It is for example possible that a PT has an edge to an USM video, but he is not able to receive audio and therefore there is no AA to the ASM audio (which an object of the USM video).

ACCESS CONNECTION ELEMENT (ACE): The ACE encompasses attributes of the B-ISDN bearer connections.

ACTION FLOW DIAGRAM: The diagram representing the action and action sequence of a (RC) entity.

ABSTRACT SERVICE MODULE (ASM): An ASM encapsulates the user's view of basic information type that may be used within a multimedia service (e.g. video, audio, data).

ALLOCATE RESOURCES: Allocating resources is the action in which reserved resources are put into action for a call.

CONFIGURATION: The configuration is the translated representation of the service request from call control, in which all implicit parts of the service request have been translated to physical resources (see physical/explicit representation).

CONSULTING A GLOBAL DATABASE: This term is introduced to describe the action in which the database, with knowledge on the total domain, is consulted. Remember this is dependent on how the resource control knowledge distribution is modeled.

DOMAIN: A domain is a combination of local domains, interconnected by interconnection trunks. A domain is the total part of a network where the RC entities have a (direct) peer-to-peer relationship with one another, and thus have (the ability to obtain) information on the resources elsewhere in the domain. This is usually the total domain of a sole network operator (administration domain), although this is not a restriction.

(SWITCHING) ENDNODES: The endnodes in the local domains connect the interconnection trunks to the other (switching) nodes in the local groups, and are thus actually gateways to the rest of the domain. It is very likely that there will be more than one endnode per local domain, because, if there is only one endnode and this node fails, communication on RC level with RC entities in other local domains has become impossible. It is important to note that all endnodes and nodes, with users connected to them (user endnodes), must support resource control.

ENTITY: An entity is an independent functional element, which can perform various functions on various controllable objects. Functional entities can be created and deleted and
communicate with higher level, lower level and peer entities. Functional entities can create lower level and peer entities.

**EXPLICIT REPRESENTATION:** An explicit way of representing the required telecommunication configuration consists of a number of objects that represent special resources and connections between the ports of the special resources and the users. Explicitly requesting implementation of telecommunication services may require more knowledge of available resources than a functional entity (call control or resource control) has. Providing the capability to implicitly specify resources empowers the client to let the server make the decision about not only which types of resources to use, but also which specific resources are required. The RC entity that handles a request will have a better idea about the availability of its own resources and hence letting this entity specify the necessary resources may result in a more efficient use of resources.

**GLOBAL DATABASE:** The global database is the central database with information on the total domain. It contains information on both the layout and the locations of all resources and users in the domain. No exact addresses of the resources are necessary (see [MOE96] for explanation).

**GLOBAL PART OF THE SERVICE REQUEST:** The global part of the service request is the part that is implemented by the RC entity at the endnodes. Here the translation to physical resources is made, by consulting the database for the availability of the necessary resources and then sending service requests (in physical representation) to the RC entities at the endnodes of the local domains where the resources are to be allocated. For the parts of a service request that belong to the global part the following rules apply:

- For point-to-point connections, there is no global part of a service request.
- For point-to-multipoint and multipoint-to-point connections all parties that belong to the same local domain are grouped into one receiving/sending group. The remaining configuration is the global part of the service request.
- For multipoint-to-multipoint connections the whole service request is global, unless at least half the parties involved in the bridging belong to the same local domain (in which case the service request is a local part for that specific local domain).

**IMPLICIT REPRESENTATION:** An implicit representation of the requested telecommunication configuration contains no references to the real implementation resources. For example, for a logical connection between two parties that use different codings a conversion is needed. Implicit representation of this configuration is possible by putting an attribute coding on the endpoints of the logical connections. The need for a converter can be derived (by the entity that controls the physical converter or an entity which knows an entity which does control the converter) by comparing the codings of the endpoints.

**INTERCONNECTION TRUNKS:** The interconnection trunks are the trunks that interconnect the local domains in a domain.

**LOCAL CALL:** A local call is a call where only parties from the same local domain are involved, and the call is set up in the same local domain (no third party call setup, with the third party belonging to another local domain than the parties actually involved in the call).
**LOCAL DATABASES**: The local databases are the databases where the information on the local domain is maintained (see Appendix 4): the User Endpoint Array, the Resource Array and the Local Nodes Array. Note that the Endnodes Interconnection Scheme Array is not considered a local database. This to make a distinction between the databases at the non-endnodes and the endnodes.

**LOCAL DOMAIN**: A local domain is a part of a total domain (a domain), in which each RC entity has (direct access to) the total knowledge of the layout of this domain, as well as the locations of all users and resources present in this domain.

**LOCAL NODE**: In the view of a specific RC entity, a local node is a node belonging to the same local domain.

**LOCAL VIEW**: The local view is the knowledge each RC entity has of the resources and users in the local domain it belongs to, as well as the interconnection scheme of this local domain.

**NON-LOCAL CALL**: A non-local call is a call where parties from different local domains are involved, or third party call setups, where at least one of the parties involved in the call does not belong to the local domain where the party, that sets up the call, belongs to.

**ORIGINATING ENDNODE**: The originating endnode is the first endnode to receive (a part of) the service request. This endnode is therefore always located in the local domain where the service request is passed on from call control to resource control. There can be more than one originating endnode per call.

**PARTY EDGE (PE)**: A PE is used to define the relationships between the PT's that may take part in a service, and the USM's that comprise the service.

**PARTY TYPE (PT)**: The party type is used to describe the different behavior of various types of parties that may be involved in a call. Each party type may have a different configuration of USM's.

**PHYSICAL REPRESENTATION**: In the physical representation the implicit model of the service request has been translated into physical resources. These resources in the representation have not been located yet and therefore do not represent actual resources within the network. This representation can be useful when only one implementation is possible, to delimit the number of translations made (a time-saving effort). The physical representation is identical to the explicit representation, with the exception that the actual address of the resources is not available. Therefore the physical representation can be transmitted in the current MAGIC interface, by using for example a dummy address for the resources. Large advantage is thus that no modifications to the MAGIC model are necessary.

---

29 Depending on the algorithm chosen.
30 An additional advantage, not present in the modeling, is that RC entities can make decisions on the configuration of calls in the network domains of other operators, without any knowledge of the locations of resources in the specified domain.
RESERVE RESOURCES: Reserving resources is the action in which resources are checked on their availability for a call and tagged if available. During the reserve phase, resources are not in use but may not be reserved for other calls.

SDL (SPECIFICATION AND DESCRIPTION LANGUAGE): A language used for the description of the action sequences of functional entities.

SERVICE MODULE (SM): The SM represents the PRM protocols\textsuperscript{31} used to implement a basic service type of a multimedia service.

SERVICE REQUEST: The service request is the representation of the service, passed on from call control to resource control. It is the task of resource control to translate it into physical resources.

TELECOMMUNICATION SERVICE LEVEL (TCS): This is a name identifying the overall telecommunication service.

USER END(POINT )NODES: User endnodes are switching nodes with users connected to them (LEX).

USER GROUP: A user group is a group of endusers from an RC entity point of view. This view may be different for RC entities belonging to other local domains. For each RC entity, the users are grouped. All users belonging to other local domains are combined into one user group per local domain, for the local domain to which the RC entity in question belongs all users on the same user endpoint node are combined into one user group. This means that the user groups are identical for all RC entities belonging to one local domain.

USER SERVICE MODULE (USM): An USM is a basic building block of service. It provides a typing function, combining one or more information types of the service into a single grouping that is understandable to the user’s view of the service (e.g. television).

\textsuperscript{31} The peer-to-peer layer protocols in the protocol reference model, see chapter 1.
Appendix 1. Call Control Local View

In this Appendix the call control local view is explained in more detail. Each CC entity has a local view of the call, except for the CC entity at the OLEX (see chapter 3), where a global view of the call is maintained. The call control local view is explained in this chapter. Examples are given of both views. More information can be found in [MAG06].

Appendix 1.1. Elementary Call Objects

An object oriented technique has been used to describe the call model. The term object is used to discuss both the global and local view model, but the term Elementary Call Object (ECO) is reserved for the local view. ECO’s are manipulated by the CC entities and are the basis of the distribution of functions among them.

Objects can be divided into three classes:

- **CALL ASSOCIATED OBJECTS** represent the aspect of associations amongst parties that are independent of other characteristics of specific telecommunications services.
- **TELECOMMUNICATIONS SERVICES ASSOCIATED OBJECTS** contain abstract attributes that define end-to-end telecommunications services, although edge-to-edge may be more appropriate, since these objects exclude local access attributes.
- **CHANNEL ASSOCIATED OBJECTS** are more directly associated with the local access attributes and mapping information flows onto the local access of a single user.

ECO’s can be dependent of other ECO’s (secondary objects, i.e. they have an owner) or may be independent, in which case they are referred to as primary objects.

Appendix 1.1.1. Control Modes

ECO’s may be divided into three subclasses based on their control mode:

- **CONFIRMED OBJECT**: A confirmed object requires concurrence between the local user and a concurring user for its construction. Any operation on a confirmed object affects its complement.
- **LOCAL OBJECT**: A local object is controlled by a single user, the beholder of the local view. If a transaction operates exclusively on local objects, its final disposition is determined without negotiations involving other users.
- **VIRTUAL OBJECTS**: An operation on a virtual object can only be directly invoked by the network. A virtual object supplements local objects or a confirmed object in a related user’s local view.
Appendix 1.2. Call Associated Objects

CAO’s can be divided into:

- **PARTY SET**: A local object, which must contain one or more (confirmed) parties to complete an association;
- **PARTY**: parties can be either confirmed or virtual.

Appendix 1.3. Telecommunications Associated Objects

TAO’s are Abstract Services. These can have Abstract Service Groups as a means of grouping. In each local view, there is a Party Service Link for each remote user associated with an Abstract Service. This Link can be confirmed or virtual, dependent of the party.

Appendix 1.4. Channel Associated Objects

ChAO’s can be divided into Upstream and Downstream Elements and Access Service Modules. Upstream Mapping Elements map the flow of information in the user to network direction, Downstream Mapping Elements in the vise versa direction. Downstream Mapping Elements can be composite (where the connection is made to the AS as a whole) or component, where distinction between the various parties in the AS can be made. An ASM encompasses a user’s endpoint of a bearer connection, encapsulating the bearer connection attributes and higher level capabilities associated with a bearer endpoint.

Appendix 1.5. Example Call Control Global and Local View Model

In Figure 1.1 a Global View of a multiparty videoconference is given. In Figure 1.2, Figure 1.3 and Figure 1.4 the corresponding Local Views of the parties involved are depicted.
Figure 1.1: Global View of a Multiparty Video Conference

Party A has a composite DME to the AS video, whereas party B has a component DME to party A in the AS and party C has component DME's to all three parties (including itself) involved in the AS.

Figure 1.2: Party A Local View of Video Conference.
Figure 1.3: Party B Local View of Video Conference.

Figure 1.4: Party C Local View of Video Conference
Appendix 2. Bearer Control Local View

In this Appendix a brief description of the bearer control services are given. The functions bearer control has to supply these services are described as well. For more information on the bearer control local view, refer to [MAG06].

The BC entities can be copied onto the functional entities in the Q.71 model (refer to [CCQ71]). The BC entities receive service requests from RC entities to make point-to-point connections. The services of bearer control (to resource control) are:

- establishment of a user plane connection.
- modification of a user plane connection.
- release of a user plane connection.

The functions BC has at its disposal are:

- establishment of BC connections.
- establishment of user plane bearer connections.
- support of the CRCG.
Appendix 3. Action Flow Diagrams for Generic Information Flows for Call Control

Appendix 3.1. Introduction

In [MAG06] the action flow diagrams for generic information flows are given. CC functional entities can be discriminated according to their function in the Q.73 model ([CCQ73]). Those related to users are referred to as CCA’s, those associated with networking as CC’s. Further discrimination can be made in terms of an entity’s role in a specific service request. The CCR model is useful in differentiating these roles.

In this Appendix the CCR model is described.

Appendix 3.2. Atomic Actions Concepts of CCR

The specifications are based on the Atomic Action concepts of CCITT X.851 Commitment, Concurrence and Recovery (CCR) (refer to [CCX851]). Atomic Actions consist of coordinated sets of operations distributed over a set of functional entities that must succeed or fail as a unit. Handling multiparty/multiconnection service requests in B-ISDN Release 2/3 and beyond makes reliable coordination of requests affecting many service components and resources crucial. An Atomic Action provides a mechanism for distributed applications, such as signaling, to develop a system with the following characteristics:

- **ATOMICITY**: A request, consisting of a set of related operations (the Atomic Action expression) on the bound data (the Call Objects), involving multiple parties, is either accepted or not (commit or rollback).
- **CONSISTENCY**: The effect of related operations in each (call control) node is performed correctly with respect to application semantics.
- **ISOLATION**: Partial results (not yet completed operations) are not accessible and do not effect other Atomic Actions.
- **DURABILITY**: Effects on operations have a lifetime beyond the Atomic Action and are not altered by node failure.

Appendix 3.3. CCR Based Information Flows
There are two general CCR based flows:

- The one-phase mechanism, where the initiator gives a READY to another node, which becomes its superior. The initiating node, which is now subordinate, waits for a COMMIT or ROLLBACK.
- The two-phase mechanism, where the initiating node remains superior by giving a BEGIN. The accepting node, which becomes a subordinate, replies with a READY or REFUSE. The superior then gives out a COMMIT or ROLLBACK.

The mechanisms can be mixed together. However, only the following situations are possible:

- From one-phase to two-phase;
- From one-phase to (a single one-phase and multiple two-phase) if all the two-phases are done before the single one-phase.

In both cases the subordinate is able to give an ABORT. However, if a COMMIT was sent already, it has precedence over the ABORT. In Figure 3.1 a few examples are depicted.

![Diagram of CCR mechanism](image)

**Figure 3.1: Example CCR cases and Atomic Action Models.**

Four major types of SDLs' are used in a telecommunications service: a cca (only in a CCA), a sup (superior), a sub (subordinate) and a coor (coordinator) state machine. Interested readers can refer to [MAG06] for more information.
Appendix 4. Distributed Resource Control Knowledge

Appendix 4.1. Introduction

In a distributed system, RC entities may have only a limited view of the domain, the users within the domain and the availability of resources within the domain. Therefore a model of the distribution of resource control knowledge is necessary. In such a model, the knowledge a specific resource control entity has of the locations of the users and the locations and the states of the resources in the domain, as well as the layout of the domain, is accurately described. For the description of the knowledge model, a network topology is necessary. The domain topology is defined in chapter 5.

Solutions for these distributed problems can be found in distributed platform solutions, such as in [BUK95], [CAR95] and [YU95]. In this Appendix five possible different models for (distributed resource control knowledge) are envisaged. The list of examples is expanded by taking a view at other possibilities, both within the network modeling (as presented in chapter 5), as without. The different models are compared on various criteria.

Criteria in the comparison of the models for distributed resource control knowledge are:

- **DATA MANAGEMENT**: How much data is stored in the domain, how much of the stored data is redundant, what are the hardware requirements to store all this data? Data here stands for the accumulated stored knowledge of the resources and users in the domain and the interconnection schemes of the domain at all RC entities.
- **HARDWARE REQUIREMENTS FOR CONNECTIONS**: What are the hardware requirements for the existing connections in the network to transport data in the network? Are new connections necessary?
- **SERVICE REQUEST MODEL TIME**: How long does it take to fulfill a service request once received from call control (both for local and interlocal calls)?
- **OPTIMALITY SEARCH**: Does the model restrict resource control in the search for an optimal configuration? The first criterion (time) is covered separately under ‘service request model time’, as the chosen model will prove to have a major impact on this criterion.

In section 4.4 an example network is given to clarify the different models presented in this Appendix.

Validation for the choice of the reference model in the thesis is given.
Appendix 4.2. Modeling Distributed Resource Control Knowledge

Appendix 4.2.1. Modeling Not Related to the Domain Topology

Modeling not within the network modeling (from chapter 5) are also taken into account to consider the possibility that the network modeling delimits the distribution of resource control knowledge.

Appendix 4.2.1.1. Model 1: Totally Distributed Resource Control

In this model each resource control entity has knowledge of the distance to each usergroup in the direction of each switching node (with resource control) it is connected to. Additionally each resource control entity has knowledge of the distances of resources in the direction of each switching (end)node (with resource control) connected to, up to a given maximum (distance).

Note that for this model the domain topology, as depicted in the previous section, has no meaning. For this model the local view is different for each RC entity, and thus terms such as local domain and endnode have no meaning.

Each resource control entity has two arrays:

**Distance Array (Array 11):**

\[
\begin{pmatrix}
A_{ij}
\end{pmatrix}
\]

- all neighbouring nodes

- all user(group)s

\[A_{ij} : \text{distance penalty to user(group) } i \text{ through node } j\]

This array represents the distances to all user(group)s when directing signalflows to them, measured through all neighboring nodes (representing all possible directions of signal flow).

**Resources Array (Array 12):**
all neighbouring nodes

all resources

\[
B_{ij}
\]

\(B_{ij}\): distance penalty for allocating resource \(i\) in direction of node \(j\) (more than one entry per matrix unit possible)

As for the previous array, this array represent the distance to a resource for each of the possible directions.

**Appendix 4.2.2. Distributed Resource Control within the Domain Topology**

In chapter 5 a domain topology was defined. In this model, each RC entity has (direct access to) an overview of the direct surrounding part of the domain, called the local domain, and the ability to obtain information on the remainder of the rest of the domain. Four different possible models are depicted in this section.

**Appendix 4.2.2.1. Model 2: Local Group Knowledge**

In this model each resource control entity has total knowledge of the locations of all resources and users present in the local domain, as well as a total interconnection scheme of all local nodes. No knowledge on the locations and availability of resources and users in other local domains is present, but can be obtained through queries between endnodes. The RC entities at the endnodes, however do have knowledge which other local domain each user(group) belongs to and have an interconnection scheme of all endnodes in the domain.
Figure 4.1: Model 2 - Local Group Knowledge.

User Endpoint Array (Array 21): all user(s) groups

\[
\begin{pmatrix}
A_j
\end{pmatrix}
\]

\(A_j\) : node to which user(group) \(j\) is connected

In this array each user(group) is related to one of the nodes (endnodes if the user(group) does not belong to the same local domain).

Resource Array (Array 22):

\[\text{For non-endnodes, only local nodes can be given. For endnodes, either local nodes or non-local endnodes, connected to, can be given.}\]
In this array all resources present at each node are given.

**Local Nodes Array (Array 23):**

\[
\begin{align*}
&\text{all local nodes} \\
&B_{ij} \\
&\text{all local nodes}
\end{align*}
\]

\(B_{ij}\): addresses of resources of type \(i\) present at node \(j\)

This array represents the total interconnection scheme of the local domain.

**Endnodes Interconnection Scheme Array (Array 24):**

\[
\begin{align*}
&\text{all endnodes} \\
&C_{ij} \\
&\text{all endnodes}
\end{align*}
\]

\(C_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.

This array represents this interconnection scheme between all endnodes.

**Appendix 4.2.2.2. Model 3: Global Access to Resource Database**

In this case each RC entity has an identical local view as in model 2. In addition each RC entity has access to a database containing global information on the exact locations of all...
users\(^2\) and resources in each local domain, as well as a total interconnection scheme of all domain nodes.

Note: How the global database is organized, is not in question here. It is for example possible that the database is split into more disjunct databases, each for a collection of local domains. These databases can exchange information, so that, from an RC entity point of view, there is global knowledge in the database it has access to. The database can also be layered, as in the model for the domain topology. The whole point is to shortly reflect on the influences of the different models of the (distributed) resource control knowledge on resource control functionality.

![Figure 4.2: Model 3 - Global Access to Resource Database.](image)

*User Endpoint Array\(^3\) (Array 31):*

\(^2\) Note that in the database all domain nodes are present. This implies that all users in the domain are grouped per user endnode.

\(^3\) See Footnote 1.
all user(group)s

\[
\begin{pmatrix}
A_j
\end{pmatrix}
\]

\(A_j\) : node to which user(group) j is connected

**Resource Array (Array 32):**

all nodes in local network

\[
\begin{pmatrix}
B_{ij}
\end{pmatrix}
\]

\(B_{ij}\) : addresses of resources of type i present at node j

**Local Nodes Array (Array 33):**

all local nodes

\[
\begin{pmatrix}
C_{ij}
\end{pmatrix}
\]

\(C_{ij}\) : distance between nodes i and j, if connected, else 0.

**Endnodes Interconnection Scheme Array (Array 34):**

all endnodes

\[
\begin{pmatrix}
D_{ij}
\end{pmatrix}
\]

\(D_{ij}\) : distance between nodes i and j, if connected, else 0.

**Database:**
all nodes

\[
\begin{pmatrix}
E_{ij}
\end{pmatrix}
\]

\(E_{ij}\) : distance between nodes i and j, if connected, else 0.

all resources

\[
\begin{pmatrix}
F_{ij}
\end{pmatrix}
\]

\(F_{ij}\) : number of resources of type i present at node j

Note: No addresses of resources are kept in the database.

all user(group)s

\[
\begin{pmatrix}
G_j
\end{pmatrix}
\]

\(G_j\) : Node to which user(group) j is connected

In the two databases all information on the layout of the total network (domain) and the locations of all users and resources is kept.

**Appendix 4.2.2.3. Model 4: Limited Access to Resource Database**

This model is identical to alternative 3, with the exception that only the RC entities at the endnodes have access to the database.
Figure 4.3: Model 4 - Limited Access to Resource Database.

User Endpoint Array\(^4\) (Array 41):

\[
( \begin{array}{c} \\
A_j \\
\end{array} )
\]

\(A_j\) : node to which user(group) \(j\) is connected

Resource Array (Array 42):

\(^4\) See Footnote 1.
Master Thesis

all nodes in local network

\[
\begin{pmatrix}
B_{ij}
\end{pmatrix}
\]

\(B_{ij}\): addresses of resources of type \(i\) present at node \(j\)

**Local Nodes Array (Array 43)**:

all local nodes

\[
\begin{pmatrix}
C_{ij}
\end{pmatrix}
\]

\(C_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.

**Endnodes Interconnection Scheme Array (Array 44)**:

all endnodes

\[
\begin{pmatrix}
D_{ij}
\end{pmatrix}
\]

\(D_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.

**Database**:

all nodes

\[
\begin{pmatrix}
E_{ij}
\end{pmatrix}
\]

\(E_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.
all nodes

\[
\begin{pmatrix}
F_{ij}
\end{pmatrix}
\]

\(F_{ij}\): number of resources of type \(i\) present at node \(j\)

all user(group)s

\[
\begin{pmatrix}
G_j
\end{pmatrix}
\]

\(G_j\): Node to which user(group) \(j\) is connected

**Appendix 4.2.2.4. Model 5: All Endnodes Having Global Information**

This alternative is identical to model 4, with the exception that in this case a local database with global knowledge is maintained at each endnode.

**Figure 4.4: Model 5 - All Endnodes Having Global Information.**
User Endpoint Array\(^5\) (Array 51):

\[
\left(\begin{array}{c}
A_j
\end{array}\right)
\]

\(A_j\): node to which user(group) \(j\) is connected

Resource Array (Array 52):

\[
\left(\begin{array}{c}
B_{ij}
\end{array}\right)
\]

\(B_{ij}\): addresses of resources of type \(i\) present at node \(j\)

Local Nodes Array (Array 53):

\[
\left(\begin{array}{c}
C_{ij}
\end{array}\right)
\]

\(C_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.

Endnodes Interconnection Scheme Array (Array 34):

\[
\left(\begin{array}{c}
D_{ij}
\end{array}\right)
\]

\(D_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.

---

\(^5\) See Footnote 1.
Database:

\[
\begin{align*}
\text{all nodes} & \quad \left( \begin{array}{c}
E_{ij} \end{array} \right) \\
\text{all resources} & \quad \left( \begin{array}{c}
F_{ij} \end{array} \right) \\
\text{all user(group)s} & \quad \left( \begin{array}{c}
G_j \end{array} \right)
\end{align*}
\]

- \(E_{ij}\): distance between nodes \(i\) and \(j\), if connected, else 0.
- \(F_{ij}\): number of resources of type \(i\) present at node \(j\).
- \(G_j\): Node to which user(group) \(j\) is connected.

Appendix 4.2.3. Alternative Models

The list of models explained in more detail in the previous sections is not exhaustive. Indeed, it is very narrow-minded, as (with the exception of model 1) they are all focused on the domains with the topology as described in chapter 5. Many other alternatives are possible, of which some will be shortly reviewed here, along with the motivation for not choosing the specific model.

Appendix 4.2.3.1. Alternative Models within the Domain Topology

- **CONJUNCT SUPER LOCAL DOMAIN DISTRIBUTED RESOURCE CONTROL**: In this alternative, the local view of each resource control entity is the same as for alternatives 2-5. The endnodes have knowledge of the layout, resources and users of the direct neighboring local domains. So the knowledge of the endnodes is identical as in model 1, with the exception that each endnode has knowledge of the layout, resources and users of the total neighboring local domains instead of just the neighboring nodes. This implies directly that, as in model 1, no guarantee can be given that RC will find an model,
even if possible\textsuperscript{6}, and this model therefore has no apparent advantages over model 2, because a RC entity at an endnode must place queries once a solution is not available in its own or one of its neighboring local domains.

• **HIERARCHICAL RESOURCE CONTROL**: In this model, the RC entities at the endnodes belong to a higher layer (from control point of view) than all other switching nodes. This has the disadvantage that an RC entity at an OLEX, in the case of a call requiring resources not available at the current local domain, must always pass the mastership of the call to another (switching) endnode, which can be disadvantageous for specific calls\textsuperscript{7}. The advantage is that an endnode has (the ability to obtain) information on all resources in the total domain. This is practically identical to model 5, and with the here above mentioned disadvantage, this model is not examined further.

Appendix 4.2.3.2. Alternative Models Not Related to the Chosen Domain Topology

Without a domain topology, not much alternatives are left. One could elaborate on the alternative depicted in model 1, but this would not yield much more result than some change in the spread of knowledge of the RC entities. Therefore, this alternative is not elaborated on further.

Appendix 4.3. Evaluation of the Alternative Models

In this section the various models from the previous sections will be evaluated on the criteria mentioned in 4. Two general remarks are necessary:

• In the comparisons the situation is considered in which the databases are updated continuously on the states of the resources (free or reserved/allocated). Whether this will eventually be implemented depends whether the hardware available can meet the requirements of continuous updating and on the demands of the network operator.

• These criteria should not be confused with the criteria for resource control functionality, as in chapter 4. The criteria here are to evaluate the different models for resource control knowledge. The influence the model has on resource control functionality (measurable by determining its influence on the criteria for resource control functionality) is one of the criteria here.

Appendix 4.3.1. Data Management

An important criterion in comparing the models is the amount of data that has to be managed, and the complexity of the hardware and software that results from this requirement for management. In this section the total size of data to be managed is calculated for each alternative. Data here stands for the accumulated stored knowledge of the resources and users in the domain and the interconnection schemes of the domain at all RC entities.

\textsuperscript{6} See 4.3.3.

\textsuperscript{7} For example where more, connections have to be made, starting at the OLEX.
VARIABLES:

\( N \): total number of users in the domain
\( n_t \): average number of users per local domain
\( n_p \): average number of users per user endpoint
\( l \): number of local domain
\( s_t \): average number of nodes with resource control per local domain
\( S \): total number of nodes with resource control
\( e \): average number of endnodes per local domain
\( r \): number of different resources
\( R \): total number of resources present in domain
\( k \): average number of neighbors per local domain node
\( e_l \): average number of endnode links per endnode (i.e. links to other endnodes)

With these variables, the average size of arrays (from the previous section) and the databases can be calculated. By multiplying these with the total number of nodes, at which these arrays and databases are kept, the total data size is obtained. \( x_{ij} \) stands for the average number of entries in the \( j^{th} \) database in alternative \( i \).

Note: For the model alternatives 2-5 the assumption is made that all switching nodes with resource control have an individual database, containing all data on the local domain the node in question belongs to. Other models are possible. One possibility is that there is one such database per local domain, with all switching nodes (with resource control) having access to it. This makes no difference in the comparisons between alternatives 2-5.

Appendix 4.3.1.1. Model 1

Distance Array:
endnodes: \[ (1+e_t+k) \left( \frac{n_t}{n_p}+(l-1) \right) = x_{11} \]
other nodes: \[ (1+k) \left( \frac{n_t}{n_p}+(l-1) \right) = x_{12} \]

Resource Array:
endnodes: \[ (1+e_t+k)r = x_{13} \]
other nodes: \[ (1+k)r = x_{14} \]
total:
\[ l\left(e(x_{11}+x_{13})+(s_t-e)(x_{12}+x_{14})\right) \]

Appendix 4.3.1.2. Model 2

User Endnode Array:
\[ \left( \begin{array}{c} n_t \varepsilon (l-1) \\ n_p \end{array} \right) = x_{21} \]

Resource Array:
\[ s_t r = x_{22} \]

Local Nodes Array:
\[ s_t^2 = x_{23} \]

Endnodes Interconnection Scheme Array:
\[ (le)^2 = x_{24} \]

Total:
\[ S(x_{21} + x_{22} + x_{23}) + le x_{24} \]

**Appendix 4.3.1.3. Model 3**

User Endnode Array:
\[ \left( \begin{array}{c} n_t \varepsilon (l-1) \\ n_p \end{array} \right) = x_{31} \]

Resource Array:
\[ s_t r = x_{32} \]

Local Nodes Array:
\[ s_t^2 = x_{33} \]

Endnodes Interconnection Scheme Array:
\[ (le)^2 = x_{34} \]

Database:
\[ S(S+r) + l \frac{n_t}{n_p} = x_{35} \]

Total:
\[ S(x_{31} + x_{32} + x_{33}) + le x_{34} + x_{35} \]

**Appendix 4.3.1.4. Model 4**

User Endnode Array:
\[ \left( \begin{array}{c} n_t \varepsilon (l-1) \\ n_p \end{array} \right) = x_{41} \]

Resource Array:
\[ s_t r = x_{42} \]

Local Nodes Array:
\[ s_t^2 = x_{43} \]

Endnodes Interconnection Scheme Array:
\[ (le)^2 = x_{44} \]

Database:
\[ S(S+r) + l \frac{n_t}{n_p} = x_{45} \]
Appendix 4.3.1.5. Model 5

User Endnode Array:
\[
\frac{n_t + (l-1)}{n_p} = x_{51}
\]

Resource Array:
\[s, r = x_{52}\]

Local Nodes Array:
\[s_t^2 = x_{53}\]

Endnodes Interconnection Scheme Array:
\[(le)^2 = x_{54}\]

Database:
\[S(S+r)+ \frac{n_t}{n_p} = x_{55}\]

Total:
\[S(x_{51} + x_{52} + x_{53}) + le(x_{54} + x_{55})\]

Appendix 4.3.1.6. Conclusions

If in the eventual algorithms for resource control the possibility is left open to check multiple configurations in the case of failed attempts, the RC entities must store information on previous, unsuccessful attempts to configure the service request (within the same service request). As this information is identical for all alternatives and as we are only interested in the differences between the alternatives this has no impact on the comparisons.

In models 1 and 2 no global database is maintained, optimizing the amount of data to be managed to local data. In model 3 and 4 the amount of data to be managed is identical, and with the global information kept in only one place, seem acceptable options from the data management point of view. In model 5, however, the global information on the layout of the network, the resources and the users is kept at all endnodes. This implies a very large database has to be kept at each of these nodes, which seems a bit exaggerate.

Appendix 4.3.2. Extra Hardware Requirements for Connections within the Domain

In this section the hardware requirements for the connections in the domain in the different models is compared. These (extra) hardware requirements are a direct result of the data that is to be managed in the network. Updating of the views is a possibility which is still considered and therefore taken into account here.

With updating of views the continuous updating of all related views of the state of resources (reserved/allocated or free) is meant.
VARIABLES:

\[ l: \] number of local domains
\[ s_i: \] average number of nodes per local domain
\[ k: \] average number of neighbors per local domain node
\[ e_i: \] total number of endnode links
\[ d_i: \] average number of calls requiring local resources in domain
\[ d_i: \] average number of calls requiring non-local resources in domain
\[ D: \] average number of calls in domain
\[ q: \] Average number of queries made (model 2)
(Note: \( D \leq d_i + d_i ! \))

Appendix 4.3.2.1. Model 1

This model requires very little extra hardware. There are only local databases to maintain the local views of the resource control entities. Updating of the views can be done by sending an update message to all direct neighbors. This imposes minimum extra information flow between the entities.

Average number of updates (per local domain)\(^9\):

\[
2k \frac{D}{l}
\]

Appendix 4.3.2.2. Model 2

The only extra hardware required are the databases at each switching point to update the local views. Updating of the local views can be done by sending an update message to all neighbors in the local domain.

Average number of updates (per local domain)\(^10\):

\[
2(s_i - 1) \frac{D}{l}
\]

Average number of queries over interconnection trunks:

\[ qd_i \]

Appendix 4.3.2.3. Model 3

This model requires a total new domain with direct access to a central database for all domain nodes. The updating and information request traffic to and from the database can become very heavy. Updating of the local views can be done by sending an update message to all neighbors in the local domain.

Average number of updates (per local domain)\(^11\):

\[ 9 \text{ Consider the average number of resource releases equal to the average number of resource allocations.} \\
10 \text{ See Footnote 9.} \\
11 \text{ See Footnote 9.} \]
Average number of database requests: 
\[ d_i \]

Average number of database updates\(^{12}\): 
\[ 2D \]

**Appendix 4.3.2.4. Model 4**

This model requires a limited new domain with direct access to a central database for all local domain endpoint nodes. The updating and information request traffic to and from the database can become very heavy. Updating of the local views can be done by sending an update message to all neighbors in the local domain.

Average number of updates (per local domain): 
\[ 2(s_i - 1) \frac{D}{l} \]

Average number of database requests: 
\[ d_i \]

Average number of database updates\(^{13}\): 
\[ 2D \]

**Appendix 4.3.2.5. Model 5**

This model requires no new domain, but large databases have to be maintained at each endnode. Besides, this imposes heavy updating traffic over the interconnection trunks.

Updating of the local views can be done by sending an update message to all neighbors in the local domain.

Average number of updates (per local domain)\(^{14}\): 
\[ 2(s_i - 1) \frac{D}{l} \]

Average number of updates sent over the interconnection trunks\(^{15}\): 
\[ 2(e_i - 1)d_i \]

**Appendix 4.3.2.6. Conclusions**

Model 1 requires almost no extra hardware and imposes a minimum extra burden on the existing trunks, and is therefore, from hardware point of view, highly preferable. Model 2 also requires hardly extra hardware, but in this model queries must be made over the interconnection trunks. In models 3 and 4 a total new domain with a large central database is required. This appears complicated, especially in model 3, where the database must be accessible from all nodes. In model 5 large databases have to be maintained at each endnode. Besides, there will be very heavy updating traffic over the interconnection trunks, delimiting

\(^{12}\) See Footnote 9.  
\(^{13}\) See Footnote 9.  
\(^{14}\) See Footnote 9.  
\(^{15}\) See Footnote 9.
the freedom for data transport over these trunks (if chosen to continually update the databases). For alternatives 3 and 4 the updating traffic to and from the central database will impose a heavy burden on the access lines to and from the database.

It is apparent that the option to continually update the databases greatly influences the hardware requirements. If the option is chosen not to continually update the databases, the differences between the various models become less apparent.

**Appendix 4.3.3. Service Request Model Time**

Another critical element in the comparison of the different models for resource control is the time it takes to configure a service request. Along with the resource reservation and allocation times of, the time it takes to update the databases plays a critical role. This is actually a criterion as mentioned in chapter 4, but the chosen way of distributing resource control knowledge directly influences this criterion in such a way, that it is mentioned separately.

For local calls, the average time for each configuration of a service request will be approximately the same. Model 1 may be slightly slower than the other four models, which are identical for local calls. Model 1 may be slower because for all other four models only one query of a local database is necessary, whereas in model 1 multiple queries may be required.

In this section for each of the five models the average call model time and the average time used for updating of the data managed are calculated. Note that these estimates are very rough, because a more accurate representation of the time aspects of the operations (at resource control level, for the setup of a call and maintenance of the data) would require a close insight into the algorithms, which, at the moment is not at hand.

**VARIABLES:**

- $t_i$: Average time for data transfer over interconnection trunks
- $t_l$: Average time used locally to setup (part of) service request or to send a message locally
- $t_d$: Average time taken to consult database
- $q$: Average number of queries made (model 2)
- $i$: Average number of other local domains involved in resource allocation

**Appendix 4.3.3.1. Model 1**

Very little can be said about the time it takes to fulfill a service request in this model, as it is not guaranteed that a service request will be fulfilled, even if the necessary resources are available in the domain.

**Appendix 4.3.3.2. Model 2**

Local call average configuration time:

footnote[16]{Considering the case where the first attempt to reserve resources yields direct success.}
Global call average configuration time:
\[ (1+i)t_i + 2(q+i)t_i \]
Time to update resource management databases:
\[ t_i \]

**Appendix 4.3.3.3. Model 3**

Local call average configuration time:
\[ t_i \]
Global call average configuration time:
\[ (1+i)t_i + 2(it_i + t_d) \]
Time to update resource management databases:
\[ t_i + t_d \]

**Appendix 4.3.3.4. Model 4**

Local call average configuration time:
\[ t_i \]
Global call average configuration time:
\[ (1+i)t_i + 2(it_i + t_d) \]
Time to update resource management databases:
\[ t_i + t_d \]

**Appendix 4.3.3.5. Model 5**

Local call average configuration time:
\[ t_i \]
Global call average configuration time:
\[ (1+i)t_i + 2(it_i) \]
Time to update resource management databases:
\[ t_i + t_i \]

**Appendix 4.3.3.6. Conclusions**

Model 1 appears to be a non-realistic option, as no guarantee can be given that an implementation of the service request, even if present in the domain, can be found. Model alternatives 2-5 are identical for local calls. The speed with which resource control can implement a service request for non-local calls in model 2 depends on the number of queries made per service and the time it takes to make these queries. By placing intelligence in resource control with which the number of queries can be minimized model 2 can become quite fast. Model 3 bears no advantage in allocation time over model 4 for non-local calls, and is therefore redundant. Model 5 is the fastest of all. The reader must bear in mind, however, that in these calculations no waiting is introduced (although this can be taken into account in the values of the time variables). Especially in models 3 and 4, where for all global
calls one database in interrogated, it is very likely that large queues will grow, especially if the option is chosen to continually update the databases.

Appendix 4.3.4. Optimality Search

Another issue in the comparison of the various models is in how far the chosen alternative delimits resource control in its effort to optimally configure a service. Optimality, as described in chapter 4, consists of three different criteria: model time, resources used and total connection costs.

For model 1, as mentioned before, no guarantee can be given that an configuration of the service request will be found, even if available. So there are cases possible where the time criterion is not met. For local calls, an optimal configuration of a service request will be found in alternatives 2 - 5, because each RC entity has a total view of the users, resources and layout of the local domain. For non-local calls, scenario 5 will use minimum time, whereas, depending on the time it takes to consult a database and to place queries, the time used in scenarios 2 - 4 will be longer. Whether the eventual configuration is optimal speaking in terms of total connection costs and resources used costs depends on how the algorithms are implemented in the RC entities at the endnodes, because that is where the decisions concerning where to allocate resources are made (the databases are checked only for the availability of the resources). No difference is distinguishable between models 2 - 5.

Appendix 4.3.4.1. Conclusions

It is clear that in model 1 it is hardly possible to find an optimal solution as only knowledge of a limited (surrounding) part of the domain is available. In models 2-5 an optimal configuration can be found for local calls. For non-local calls, model 5 will prove optimal concerning the time criterion, whereas in alternatives 2 - 4 the time taken will be longer (depending on the time it takes to consult a database and to place queries). For all alternative models 2 - 5, the optimality achievable concerning total connection costs and resources used costs depends on how the algorithms are implemented in resource control and is thus not restricted by the alternative chosen.
Appendix 4.4. Example Network to Clarify Models for Distributed Resource Control Knowledge

Figure 4.5: Layout of the example network.
In Figure 4.5 an example network is depicted, to further clarify the model for distributed resource control knowledge. The roman numbered circles represent local networks. The small circles are nodes in the local network, the gray ones with resource control. The dotted lines depict the resource control entity peer-to-peer relations, the thin black lines the complete interconnection scheme in the local network. The thick black lines represent the interconnection trunks. The distance between resource control nodes can be obtained by counting the number of hops (thus distance here represents the number of hops). For the interconnection trunks the length (in hops) is given also. In Table 1 a (fictive) distribution of three sample resources in the local network I is given.

### Table 1: Resources available array for example network

<table>
<thead>
<tr>
<th></th>
<th>Converter</th>
<th>Distributer_2</th>
<th>Distributer_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Appendix 4.4.1. Model 1:

#### Distance Array for Node A

<table>
<thead>
<tr>
<th>to</th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>H</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

#### Resource Array for Node A

<table>
<thead>
<tr>
<th>resource</th>
<th>via</th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converter</td>
<td></td>
<td>1,2</td>
<td>3,3,3</td>
<td>2,2</td>
<td>3,3</td>
</tr>
<tr>
<td>Distributer_2</td>
<td></td>
<td>2,2,3</td>
<td>2,2,3</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>Distributer_3</td>
<td></td>
<td>2,2,3</td>
<td>2,2,3</td>
<td>-</td>
<td>3</td>
</tr>
</tbody>
</table>

- Note: The maximum distance for resources to be entered in the resource array is 3.

### Appendix 4.4.2. Models 2-5:
User Endpoint Array:

<table>
<thead>
<tr>
<th>usergroup</th>
<th>A</th>
<th>B</th>
<th>D</th>
<th>E</th>
<th>H</th>
<th>II</th>
<th>III</th>
<th>IV</th>
<th>V</th>
<th>VI</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>E</td>
<td>H</td>
<td>F</td>
<td>F</td>
<td>G</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

Resource Array (partly):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>con</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>dis_2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>dis_3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Local Nodes Array:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Local nodes interconnection scheme array:

<table>
<thead>
<tr>
<th>F</th>
<th>G</th>
<th>IIx</th>
<th>IIIx</th>
<th>IVx</th>
<th>Vx</th>
<th>VIx</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>18</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: For the local networks II - VI there is only one local endpoint; IIx - VIx.

Note: If the connection between nodes F and G is given in both the Local Nodes Array and the Local Nodes Interconnection Scheme Array, there is some redundant information present. This will in general not be a problem, as the number of endnodes per local network is expected to remain small. It is maintained in both arrays for simplicity.

For the database identical arrays as the User endpoint, Resource and Local Nodes Array can be defined, with the only difference being that all nodes present in the network are included.
Appendix 4.5. Conclusions

In this Appendix different models for the distribution of RC knowledge are presented. The majority of the models focuses on the network model presented in chapter 5. Five different models are compared. Factors that play a role in the comparison are software and hardware requirements and the effect the model might have on the criteria for resource control (see chapter 4), especially the time used for the setup of a service request.

Which of the different models of distributed resource control knowledge is preferable is highly dependent on the criteria evaluated and how these criteria are rated. As the criteria need input which must be obtained empirically, testruns are needed to come to a decisive comparison. Alternative 4 (knowledge in the local domain, knowledge on other local domains accessible at endnodes through a central database) is chosen as a reference model in the thesis, because:

- The soft- and hardware requirements seem acceptable.
- The model does not delimit resource control in its functionality (that is search for the optimal translation of a service request).
- The time consumption of the model is dependent of the implementation of the database and therefore can be reduced to an absolute minimum.

It is important to note that the model is used as a reference model, and all other alternatives can replace it by redefining the network.
Appendix 5. Syntax Rules for the CC-RC and RC-RC Interfaces

In this Appendix the syntax descriptions of both the CC-RC and RC-RC interfaces are described.

Appendix 5.1. Syntax Rules for the CC-RC interface

Figure 5.1: CC-RC Interface

```
<CC-RC primitive description> ::= (normal primitive type)(aa_params, <parameter list>)
| (return primitive type)(aa_params, <return parameter list>)
| (info primitive type)(info_params)
| (error primitive type)(aa_params, (reason, <reason>))

<normal primitive type> ::= BEGIN

<return primitive type> ::= READY
            | COMMIT

<info primitive type> ::= INFO

<error primitive type> ::= REFUSE
            | ROLLBACK
            | ABORT

aa_params = aa_id, sender, senderaptype, receiver, receiveraptype, info

info_params = VCI, <bearer control params>
```
Appendix 5.2. Syntax Rules for the RC-RC Interface
<RC-RC primitive description> ::= 
{<normal primitive type>:(<aa_params>, <parameter list>)}
{<return primitive type>:(<aa_params>, <return parameter list>)}
{<error primitive type>:(<aa_params>, (reason, <reason>))}
{<query primitive type>:(query_params)}

<normal primitive type> ::= BEGIN
INFO

$return primitive type$ ::= READY
COMMlT

$error primitive type$ ::= REFUSE
ROLLBACK
ABORT

$query primitive type$ ::= AVAILABLE
POSREPLY
NEGREPLY

$aa_params ::= $ aa_id, sender, senderaptype, receiver, receiveraptype, info

$query_params ::= <$implicit object ref>$<$explicit object ref>

$parameter list ::= $ [{<operation name>:(<implicit object ref>$<explicit object ref>)}
{<operation name>:(<implicit object ref>$<explicit object ref>)}*

$return parameter list ::= $ <operation name>:(<implicit object ref>$<explicit object ref>)}
{<operation name>:(<implicit object ref>$<explicit object ref>)}*

$operational mode ::= $ MIO

$operation list1 ::= $ {<operation name>:(<e_Endpoint>, <e_endpoint ref>,

Master Thesis

(continued text)

<e_endpoint attr parameter list>
|<operation name>(<i_bridge ref>, <i_distributor ref>,
 |<i_combiner ref>, <i_connection ref>)
|<operation name>(<i_Distributor, <i_distributor ref>,
 |<i_combiner attr parameter list>)
|<operation name>(<i_Combiner, <i_combiner ref>,
 |<i_connection attr parameter list>)
|<operation name>(<i_Connection, <i_connection ref>,
 |<i_connection attr parameter list>)
|<operation name>(<i_Sync group, <i_sync group ref>,
 |<i_sync group attr parameter list>)
|<operation name>(<e_Bridge, <e_bridge ref>,
 |<e_bridge attr parameter list>)
|<operation name>(<e_Distributor, <e_distributor ref>,
 |<e_distributor attr parameter list>)
|<operation name>(<e_Combiner, <e_combiner ref>,
 |<e_combiner attr parameter list>)
|<operation name>(<e_Connection, <e_connection ref>,
 |<e_connection attr parameter list>)
|<operation name>(<e_Converter, <e_converter ref>,
 |<e_converter attr parameter list>)
|<operation name>(<e_Routing group, <e_routing group ref>,
 |<e_routing group attr parameter list>)

<operation list2> ::= { delete((<implicit object ref>,<explicit object ref>)) }

<operation name> ::= createlmodify

<implicit object ref> ::= <i_bridge ref>l<i_distributor ref>l<i_combiner ref>

<explicit object ref> ::= <e_endpoint ref>l<e_bridge ref>l<e_distributor ref>

<e_endpoint attr parameter list> ::= <address value>, { <port ref>(<input>l<output>,
 |<asm_coding value») }, <port ref>,
 |<input>l<output>, <asm_coding value»} *

<i_bridge attr parameter list> ::= <abstract service id>, <inport ref list>
 |(<port ref>{<signal_format>, <combining algorithm>}),
 |<time_threshold_value>, <connection_threshold_value>,
 |<resources_threshold_value>

<i_distributor attr parameter list> ::= <abstract service id>, <inport ref>, <outport ref list>,
 |<time_threshold_value>, <connection_threshold_value>,
 |<resources_threshold_value>

<i_combiner attr parameter list> ::= <abstract service id>, <inport ref list>, <outport ref>({<signal_format>,
 |<combining algorithm>}), <time_threshold_value>,
 |<connection_threshold_value>, <resources_threshold_value>

<i_connection attr parameter list> ::= <abstract service id>, <inport ref>, <outport ref>, <time_threshold_value>,
 |<connection_threshold_value>, <resources_threshold_value>

<i_sync group parameter list> ::= <abstract service id list>
Appendix 5.3. Syntax Rules for the RC-BC Interface

Figure 5.3: The RC-BC Interface.
ITHROUGHCONNECT

<parameter list> ::=  
{<BEARER_SETUP.req.ind parameter list>}
|{<BEARER_SETUP.resp.conf parameter list>}
|{<BEARER_SETUP.REJ.req.ind parameter list>}
|{<BEARER_RELEASE.req.ind parameter list>}
|{<BEARER_RELEASE.resp.conf parameter list>}
|{<BEARER_MODIFY.req.ind parameter list>}
|{<BEARER_MODIFY.resp.conf parameter list>}
|{<BEARER_MODIFY.REJ.req.ind parameter list>}
|{<RESET.req.ind parameter list>}
|{<INFO.req parameter list>}
|{<INFO.conf parameter list>}
|{<ITHROUGHCONNECT parameter list>}

<BEARER_SETUP.req.ind parameter list> ::=  
{<call ref>, <bearer control ref>, <destination address>,
[<connection element ref>], [<common route ref>], <sender endpoint ref>, <receiver endpoint ref>, <atmpar values>}

<BEARER_SETUP.resp.conf parameter list> ::=  
{<call ref>, <bearer control ref>, [<connection element ref>],
<sender endpoint ref>, <receiver endpoint ref>}

<BEARER_SETUP.REJ.req.ind parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <cause indicator value>}

<BEARER_RELEASE.req.ind parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <cause indicator value>}

<BEARER_RELEASE.resp.conf parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>}

BEARER_MODIFY.req.ind parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <atmpar values>, [<routing info request>]

BEARER_MODIFY.resp.conf parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, [<routing info list>]

BEARER_MODIFY.REJ.req.ind parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <cause indicator value>}

<RESET.req.ind parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <cause indicator value>}

<INFO.req parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <cause indicator value>}

<INFO.conf parameter list> ::=  
{<call ref>, <bearer control ref>, <sender endpoint ref>,
<receiver endpoint ref>, <routing info request>}

<THROUGHCONNECT parameter list> ::=  
{<call ref>, <connection element ref>}

<routing info request> ::=  
RoutingInformationRequest

<routing info list> ::=  
{<node address ref>{,<node address ref>}*

<atmpar values> ::=  
{<forward values>, <backward values>
<forward values> ::= <cell delay variation ref>, <cell loss rate ref>, <mean bandwidth ref>, <peak bandwidth ref>

<backward values> ::= <cell delay variation ref>, <cell loss rate ref>, <mean bandwidth ref>, <peak bandwidth ref>
Appendix 6. ASN.1 Descriptions of Resource Control Interface Elements

Appendix 6.1. Introduction

One of the fundamental problems in data communications is dealing with the diverse formats used to represent information in different systems. At the programming level, applications in different systems may wish to represent complex data structures in different ways and still retain the ability to exchange these data structures. Abstract Syntax Notation One (ASN.1), together with the Basic Encoding Rules, provides a system-independent way to describe and represent a broad range of data, and is a fundamental part of the open system interconnection solution to the problem of different data representations.

Representing information abstractly is not enough for data communication: an encoding is necessary as well. For this, a set of encoding rules that generate an unambiguous (bit-level) representation of the data to be described is necessary. In International Standards 8824 and 8825 ASN.1 and the Basic Encoding Rules, which can be used as a transfer syntax, are described. The term ASN.1 is often used to describe the combination of them.

In this Appendix a short description of the ASN.1 language is given and the ASN.1 descriptions of the RC peer-to-peer, as well as for the CC-RC interface, information transport are described. With these, automated tools (such as SNACC) can be used to auto-generate C language.

Appendix 6.2. The ASN.1 Language

ASN.1 has many similarities to the data type definition aspects of such conventional programming languages as Pascal, C and Ada, or even older languages. Each of these, like ASN.1, has a formal syntax for defining and naming types as well as mechanisms that allow users to build up arbitrarily complex types out of simpler preexisting types.

With the ASN.1 description language data structures and fundamental operations on them can be described independently of the eventual programming language. For a more detailed description of the ASN.1 descriptions, please refer to [CHA89], [NEU92] and [ASN94].

Appendix 6.3. The RC Peer-to-Peer Communication
The gray shaded areas denote the descriptions, necessary for the queries between RC entities.

```
RCRCPrimitiveDescription DEFINITIONS IMPLICIT TAGS ::= BEGIN

RCRCPrimitiveDescription ::= CHOICE
  { normalprimitive [1] NormalPrimitive,
    returnprimitive [2] ReturnPrimitive,
    errorprimitive [3] ErrorPrimitive,
    queryprimitive [4] QueryPrimitive
  }

NormalPrimitive ::= SEQUENCE
  { normalprimitivetype [1] NormalPrimitiveType,
    aaparams [2] AAParams,
    parameterlist [3] ParameterListSequence
  }

ReturnPrimitive ::= SEQUENCE
  { returnprimitivetype [1] ReturnPrimitiveType,
    aaparams [2] AAParams,
    returnparameterlist [3] ReturnParameterListSequence
  }

ErrorPrimitive ::= SEQUENCE
  { errorprimitivetype [1] ErrorPrimitiveType,
    aaparams [2] AAParams,
    reason [3] Reason
  }

QueryPrimitive ::= SEQUENCE
  { queryprimitivetype [1] QueryPrimitiveType,
    queryparameters [2] QueryParameters
  }

NormalPrimitiveType ::= ENUMERATED
  { begin (1),
    info (2)
  }

Note: See Note in (chapter 6)

ReturnPrimitiveType ::= ENUMERATED
  { ready (1),
    commit (2)
  }

ErrorPrimitiveType ::= ENUMERATED
  { refuse (1),
    rollback (2),
    abort (3)
  }

QueryPrimitiveType ::= ENUMERATED
  { available (1),
    postreply (2),
    negreply (3)
  }
```
AAParams ::= SEQUENCE
{ aaid [1] AAId,
sender [2] Sender,
senderaptype [3] SenderApType,
receiver [4] Receiver,
receiveraptype [5] ReceiverApType,
info [6] Info }

QueryParameters ::= SEQUENCE
{ objectrefchoice [1] ObjectRefChoice }

ParameterListSequence ::= SEQUENCE OF ParameterList

ParameterList ::= SEQUENCE
{ operationalmode [1] OperationalMode,

ReturnParameterListSequence ::= SEQUENCE OF ReturnParameterList

ReturnParameterList ::= SEQUENCE
{ operationname [1] OperationName,
  objectrefchoice [2] ObjectRefChoice }

OperationalMode ::= ENUMERATED
{ mandatory (1),
  optional (2) }

OperationListChoice ::= CHOICE
{ operationlist1 [1] OperationListOne,
  operationlist2 [2] OperationListTwo }

OperationListOne ::= CHOICE
{ eendpoint [1] EEndpoint,
  ibridge [2] IBridge,
  icombiner [3] ICombiner,
  icompression [4] ICompression,
  idistributor [5] IDistributor,
  isyncgroup [6] ISyncGroup,
  ebridge [7] EBridge,
  ecombiner [8] ECombiner,
  econversion [9] EConversion,
  econverter [10] EConverter,
  edistributor [11] EDistributor,
  eroutegroup [12] ERoutingGroup }

OperationListTwo ::= SEQUENCE
{ objectrefchoice [1] ObjectRefChoice }

EEndpoint ::= SEQUENCE
{ operationname [1] OperationName,
eendpointref [2] EEndpointRef, }
eendpointattributes [3] EEndpointAttrParameterList

IBridge ::= SEQUENCE
  { operationname [1] OperationName,
    ibridgeref [2] IBridgeRef,
    ibridgeattributes [3] IBridgeAttrParameterList
  }

ICombiner ::= SEQUENCE
  { operationname [1] OperationName,
    icombinerref [2] ICombinerRef,
    icombinerattributes [3] ICombinerAttrParameterList
  }

IConnection ::= SEQUENCE
  { operationname [1] OperationName,
    iconnectionref [2] IConnectionRef,
    iconnectionattributes [3] IConnectionAttrParameterList
  }

IDistributer ::= SEQUENCE
  { operationname [1] OperationName,
    idistributerref [2] IDistributerRef,
    idistributerattributes [3] IDistributerAttrParameterList
  }

ISyncGroup ::= SEQUENCE
  { operationname [1] OperationName,
    isyncgroupref [2] ISyncGroupRef,
    isyncgroupattributes [3] ISyncGroupAttrParameterList
  }

EBridge ::= SEQUENCE
  { operationname [1] OperationName,
    ebridgeref [2] EBridgeRef,
    ebridgeattributes [3] EBridgeAttrParameterList
  }

ECombiner ::= SEQUENCE
  { operationname [1] OperationName,
    ecombinerref [2] ECombinerRef,
    ecombinerattributes [3] ECombinerAttrParameterList
  }

EConnection ::= SEQUENCE
  { operationname [1] OperationName,
    econnectionref [2] EConnectionRef,
    econnectionattributes [3] EConnectionAttrParameterList
  }

EConverter ::= SEQUENCE
  { operationname [1] OperationName,
    econverterref [2] EConverterRef,
  }

EDistributer ::= SEQUENCE
  { operationname [1] OperationName,
Master Thesis

edistributorref [2] EDistributorRef,

ERoutingGroup ::= SEQUENCE
{ operationname [1] OperationName,
eroutinggroupref [2] ERoutingGroupRef,
eroutinggroupattributes [3] ERoutingGroupAttrParameterList
}

OperationName ::= ENUMERATED
{ create (1), modify (2) }

ObjectRefChoice ::= CHOICE
{ implicitobjectref [1] ImplicitObjectRef,
explicitobjectref [2] ExplicitObjectRef }

ImplicitObjectRef ::= ENUMERATED
{ ibridgeref (1), icombinerref (2),
iconnectionref (3), idistributerref (4),
isyncgroupref (5) }

ExplicitObjectRef ::= ENUMERATED
{ ebridgeref (1), ecombinerref (2),
econnectionref (3), econverterref (4),
edistributerref (5), eendpointref (6),
eroutinggroupref (7) }

EEndpointAttrParameterList ::= SEQUENCE
{ addressvalue [1] AddressValue,
portlist [2] PortList }

IBridgeAttrParameterList ::= SEQUENCE
{ abstractserviceid [1] AbstractServiceId,
inputportreflist [2] PortRefList,
outputportlist [3] OutputPortOneList,
timethresholdvalue [4] TimeThresholdValue,
maxnumberofparties [5] MaxNumberOfParties,
connectionthresholdvalue [6] ConnectionThresholdValue,
resourcethresholdvalue [7] ResourceThresholdValue }

ICombinerAttrParameterList ::= SEQUENCE
{ abstractserviceid [1] AbstractServiceId,
inputportreflist [2] PortRefList,
outputport [3] OutputPortOne,
timethresholdvalue [4] TimeThresholdValue,
maxnumberofparties [5] MaxNumberOfParties,
connectionthresholdvalue [6] ConnectionThresholdValue,  
resourcethresholdvalue [7] ResourceThresholdValue  
}

IConnectionAttrParameterList ::= SEQUENCE  
{  
abstractserviceid [1] AbstractServiceId,  
inputportref [2] PortRef,  
outputportref [3] PortRef,  
timethresholdvalue [4] TimeThresholdValue,  
connectionthresholdvalue [5] ConnectionThresholdValue,  
}

IDistributerAttrParameterList ::= SEQUENCE  
{  
abstractserviceid [1] AbstractServiceId,  
inputportref [2] PortRef,  
outputportreflist [3] PortRefList,  
timethresholdvalue [4] TimeThresholdValue,  
connectionthresholdvalue [5] ConnectionThresholdValue,  
}

ISyncGroupAttrParameterList ::= SEQUENCE OF AbstractServiceId  

EBridgeAttrParameterList ::= SEQUENCE  
{  
asmcodingvalue [1] AsmCodingValue,  
addressvalue [2] AddressValue,  
inputportreflist [3] PortRefList,  
outputportref [4] PortRefTwoList  
}

ECombinerAttrParameterList ::= SEQUENCE  
{  
asmcodingvalue [1] AsmCodingValue,  
addressvalue [2] AddressValue,  
inputportreflist [3] PortRefList,  
outputport [4] OutputPortTwo  
}

EConnectionAttrParameterList ::= SEQUENCE  
{  
asmcodingvalue [1] AsmCodingValue,  
inputportref [2] PortRef,  
outputportref [3] PortRef  
}

EConverterAttrParameterList ::= SEQUENCE  
{  
addressvalue [1] AddressValue,  
inputportref [2] PortRef,  
inputportasmcodingvalue [3] AsmCodingValue,  
outputportref [4] PortRef,  
outputportasmcodingvalue [5] AsmCodingValue  
}

EDistributerAttrParameterList ::= SEQUENCE  
{  
asmcodingvalue [1] AsmCodingValue,  
addressvalue [2] AddressValue,  
inputportref [3] PortRef,  
outputportreflist [4] PortRefList  
}
ERoutingGroupAttrParameterList ::= SEQUENCE OF EConnectionRef

PortList ::= SEQUENCE OF Port

Port ::= SEQUENCE
  { portref [1] PortRef,
    porttype [2] PortType,
    asmcodingvalue [3] AsmCodingValue
  }

PortType ::= ENUMERATED
  { input (1),
    output (2)
  }

PortRefList ::= SEQUENCE OF PortRef

OutputPortOneList ::= SEQUENCE OF OutputPortOne

OutputPortOne ::= SEQUENCE
  { outputportref [1] PortRef,
    signalformat [2] SignalFormat,
    combiningalgorithm [3] CombiningAlgorithm
  }

SignalFormat ::= CHOICE
  { composit [1] NULL,
    component [2] Component
  }

Component ::= SEQUENCE
  { accesslist [1] PortRefList,
    maxnumberofparties [2] INTEGER OPTIONAL
  }

OutputPortTwoList ::= SEQUENCE OF OutputPortTwo

OutputPortTwo ::= SEQUENCE
  { outputportref [1] PortRef,
    combiningalgorithm [3] CombiningAlgorithm
  }

--no further specification:

AAId ::= INTEGER

Sender ::= OCTET STRING

SenderApType ::= OCTET STRING

Receiver ::= OCTET STRING

ReceiverApType ::= OCTET STRING

Info ::= OCTET STRING

PortRef ::= OCTET STRING

CombiningAlgorithm ::= OCTET STRING
Appendix 6.4. The CC-RC Interface

CCRCPrimitiveDescription DEFINITIONS IMPLICIT TAGS ::= BEGIN

CCRCPrimitiveDescription ::= CHOICE {
  normalprimitive [1] NormalPrimitive,
  returnprimitive [2] ReturnPrimitive,
  errorprimitive [3] ErrorPrimitive,
  infoprimitive [4] InfoPrimitive
}

normalprimitive ::= SEQUENCE { aaaparams [1] AAParams, }
parameterlist [2] ParameterListSequence

ReturnPrimitive ::= SEQUENCE
{ returnprimitivetype [1] ReturnPrimitiveType,
aaparams [2] AAParams,
returnparameterlist [3] ReturnParameterListSequence
}

ErrorPrimitive ::= SEQUENCE
{ errorprimitivetype [1] ErrorPrimitiveType,
aaparams [2] AAParams,
reason [3] Reason
}

InfoPrimitive ::= SEQUENCE
{ infoparameters [1] InfoParameters
}

ReturnPrimitiveType ::= ENUMERATED
{ ready (1),
  commit (2)
}

ErrorPrimitiveType ::= ENUMERATED
{ refuse (1),
  rollback (2),
  abort (3)
}

AAParams ::= SEQUENCE
{ aaid [1] AAId,
sender [2] Sender,
senderaptype [3] SenderApType,
receiver [4] Receiver,
receiveraptype [5] ReceiverApType,
info [6] Info
}

InfoParameters ::= SEQUENCE
{ vci [1] VCI,
bearercontrolparameters [2] BearerControlParameters
}

ParameterListSequence ::= SEQUENCE OF ParameterList

ParameterList ::= SEQUENCE
{ operationalmode [1] OperationalMode,
}

ReturnParameterListSequence ::= SEQUENCE OF ReturnParameterList

ReturnParameterList ::= SEQUENCE
{ operationname [1] OperationName,
  objectrefchoice [2] ObjectRefChoice
}

OperationalMode ::= ENUMERATED

137
{ mandatory (1),
   optional (2)
 }

OperationListChoice ::= CHOICE
 { operationlist1 [1] OperationListOne,
   operationlist2 [2] OperationListTwo
 }

OperationListOne ::= CHOICE
 { endpoint [1] EEndpoint,
   ibridge [2] IBridge,
   isyncgroup [3] ISyncGroup
 }

OperationListTwo ::= SEQUENCE
 { objectrefchoice [1] ObjectRefChoice
 }

EEndpoint ::= SEQUENCE
 { operationname [1] OperationName,
   eendpointref [2] EEndpointRef,
   eendpointattributes [3] EEndpointAttrParameterList
 }

IBridge ::= SEQUENCE
 { operationname [1] OperationName,
   ibridgeref [2] IBridgeRef,
   ibridgeattributes [3] IBridgeAttrParameterList
 }

ISyncGroup ::= SEQUENCE
 { operationname [1] OperationName,
   isyncgroupref [2] ISyncGroupRef,
   isyncgroupattributes [3] ISyncGroupAttrParameterList
 }

OperationName ::= ENUMERATED
 { create (1),
   modify (2)
 }

ObjectRefChoice ::= CHOICE
 { implicitobjectref [1] ImplicitObjectRef,
   explicitobjectref [2] ExplicitObjectRef
 }

ImplicitObjectRef ::= ENUMERATED
 { ibridgeref (1),
   isyncgroupref (2)
 }

ExplicitObjectRef ::= EEndpointRef

EEndpointAttrParameterList ::= SEQUENCE
 { addressvalue [1] AddressValue,
   portlist [2] PortList
 }

Distributed Resource Control for B-ISDN Release 2/3

(1), (2)
IBridgeAttrParameterList ::= SEQUENCE
  { abstractserviceid [1] AbstractServiceId,
    inputportreflist [2] PortRefList,
    outputportlist [3] OutputPortOneList,
  }

ISyncGroupAttrParameterList ::= SEQUENCE OF AbstractServiceId

PortList ::= SEQUENCE OF Port

Port ::= SEQUENCE
  { portref [1] PortRef,
    porttype [2] PortType,
    asmcodingvalue [3] AsmCodingValue
  }

PortType ::= ENUMERATED
  { input (1),
    output (2)
  }

PortRefList ::= SEQUENCE OF PortRef

OutputPortOneList ::= SEQUENCE OF OutputPortOne

OutputPortOne ::= SEQUENCE
  { outputportref [1] PortRef,
    signalformat [2] SignalFormat,
    combiningalgorithm [3] CombiningAlgorithm
  }

SignalFormat ::= CHOICE
  { composit [1] NULL,
    component [2] Component
  }

Component ::= SEQUENCE
  { accesslist [1] PortRefList,
    maxnumberofparties [2] INTEGER OPTIONAL
  }

--no further specification:

AAId ::= INTEGER

Sender ::= OCTET STRING

SenderApType ::= OCTET STRING

Receiver ::= OCTET STRING

ReceiverApType ::= OCTET STRING

Info ::= OCTET STRING

PortRef ::= OCTET STRING

CombiningAlgorithm ::= OCTET STRING
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractServiceId</td>
<td>INTEGER</td>
</tr>
<tr>
<td>AsmCodingValue</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>AddressValue</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>EEndpointRef</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>IBridgeRef</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>ISyncGroupRef</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>TimeThresholdValue</td>
<td>INTEGER</td>
</tr>
<tr>
<td>MaxNumberOfParties</td>
<td>INTEGER</td>
</tr>
<tr>
<td>VCI</td>
<td>INTEGER</td>
</tr>
<tr>
<td>BearerControlParameters</td>
<td>OCTET STRING</td>
</tr>
<tr>
<td>Reason</td>
<td>OCTET STRING</td>
</tr>
</tbody>
</table>
Appendix 7. Proof of Limitations of MAGIC Deliverable 6 Adaptation Function for CC-RC Interface

In this Appendix, an example is given to prove that the adaptation function for translating the call control local view model into a resource control service request representation makes certain assumptions on resources. This implies that resource control is restricted in translation of the service request, or has to enhance the set of translation rules.

Figure 7.1: Implicit configuration derived from call control local view model with adaptation function as presented in MAGIC deliverable 6, section 5.3.5.

Figure 7.2: Alternative solution.

If resource control were to translate a service request, received as in Figure 7.1, translation to the model in Figure 7.2 would not be possible without alternation of the rules.
Appendix 8. Correction of Figure S5.54 ([MAG06])

In [MAG06] the instance connections for the implicit objects is given on page 88 (figure S5.54). As a result of the corrections (see chapter 6) the model has changed. The corrected model is depicted in Figure 8.1.

Figure 8.1: Instance Connections for implicit objects.

Appendix 9.1. Introduction

In this Appendix algorithms are given for the translation of a service request to physical resources. In section 9.2 algorithms are given for point-to-point, point-to-multipoint and multipoint-to-point connections. These algorithms optimize the configuration of a service request in terms of resources used and connection costs. For multipoint-to-multipoint connections with component downstream mapping (see Appendix 1), the complexity of the problem is described. In section 9.3 some heuristic algorithms are given which require less time in finding an (acceptable) configuration.

Appendix 9.2. Algorithms Optimizing Configuration

In this section a number of alternative algorithms are given, which optimize the configuration in terms of resources used and total connection costs.

Appendix 9.2.1. Point-to-Point Connection: Implicit Connection

Appendix 9.2.1.1. Complexity of the problem

The problem concerning the configuration of an implicit connection is more complex than one can imagine by taking a short glance at the problem. Every cascade of (at least 0) converters is a possible configuration, as long as the end-to-end conversion of the cascade equals the required conversion. The optimal configuration depends on the costs of the converters and the costs for the connections, which carry the various possible user plane signals with different bandwidths. In this Appendix an algorithm is introduced, which finds the optimal solution in $O(R)$ time, with $R$ the number of codings possible.

Appendix 9.2.1.2. Syntax

i_Connection ( IConn, Reference )
For all conversions from coding = \( J_l \) to coding = \( J_R \), create a list of \( C_x(J_i, J_j) \), \( i, j = 1, \ldots, R \) (\( i \neq j \)), with: \( \forall x \in \{1, \ldots, M-1\} \) \( R(C_x(J_i, J_j)) < R(C_{x+1}(J_i, J_j)) \)\(^{17}\).

The eventual configuration will consist of a cascade of converters (minimal 0 or 1, depending on coding_A and coding_B), notated as:

\[ I_{(a,b)} = (i_1, p_1, j_1, i_2, p_2, j_2, \ldots, j_N, i(N+1), p(N+1)) = (C_{i_1}(coding_A, J_{j_1}), p_1, C_{i_2}(J_{j_1}, J_{j_2}), p_2, \ldots, C_{i(N+1)}(J_{j_N}, coding_B), p(N+1)) \]

Here:

- \( J_{j_1} \ldots J_{j(N+1)} \) The successive codings within the connection
- \( i_1 \) The configuration number of the resource (\( i \) is optimal, i.e. minimum costs, etc.).
- \( p_i \) The location for the converter in this sequence. \( N(p) \) is the successive optimal location following location \( p \).
  - \( \forall_{i,j} ((i \neq j) \Rightarrow (P(p_i) \neq P(p_j))) \) within a sequence
  - \( P(p_i) \in \{<Network>\} \)
  - \( n(p) \): the introduction sequence number. \( n(p) = 1 \) implies that \( p \) is the first introduced position and thus the optimal, etc.

\[ I_{(a,b)} = (i_1, p_1, j_1, i_2, p_2, j_2, \ldots, j_N, i(N+1), p(N+1)) \]

The unique identifier of this sequence

- \( \text{amax} \) MAX\((a)\)
- \( \text{bmax}(a) \) MAX\((b)\) for this specific value of \( a \).

The cost for the configurations is a function of the elements of the sequence, identifying the configuration:

\[ \text{Cost}(I_{(a,b)}) = \text{Cost}(i_1, p_1, j_1, i_2, p_2, j_2, \ldots, j_N, i(N+1), p(N+1)) = \text{Cost}(\text{Connection}(I_{a,b})) + \text{Cost}(\text{Resources}(I_{a,b})) \]

\[ \text{Cost}(\text{Resources}(I_{(a,b)})) = R(C_{i_1}(coding_A, J_{j_1})) + R(C_{i_2}(J_{j_1}, J_{j_2})) + \ldots + R(C_{i(N+1)}(J_{j_N}, coding_B)) \]

\[ \text{Cost}(\text{Connection}(I_{(a,b)})) = b(J_1)d(P(p_1), P(p_2)) + b(J_2)d(P(p_2), P(p_3)) + \ldots + b(J_N)d(P(p_n), P(p_{n+1})) + b(coding_A)d(P(A), P(p_1)) + b(coding_B)d(P(p_{n+1}), P(B)) \]

If the bandwidth can be optimized, that is, the full bandwidth is not required for the connection (in the case of composit downstream mapping), this is included in the codings of the endpoints.

\(^{17} M \) is the number of possible conversions \( C_x(J_i, J_j) \).
Appendix 9.2.1.3. Introduce Initial Alternative Configurations

If \( \text{coding}_A = \text{coding}_B \) then alternative \( I_{(0,1)} = () \) else alternative \( I_{(0,1)} = (1,1) = (C_i(\text{coding}_A, \text{coding}_B), 1) \).

For all \( i \) with \( J_i \not\in \{ \text{coding}_A, \text{coding}_B \} \) introduce alternatives:

\[
I_{(i,1)} = (1, i, 1, 1, 1) = (C_i(\text{coding}_A, J_i), 1, C_i(\text{coding}_J, B), 1).
\]

The optimal location of the converters depends on the bandwidth of \( \text{coding}_A, J_i \) and \( \text{coding}_B \). It is clear that only alternatives with \( (b(J_i) < b(\text{coding}_A)) \text{ and } (b(J_i) < b(\text{coding}_B)) \) are taken into account first\(^\text{18}\).

Appendix 9.2.1.4. Algorithm

Take the alternative introducing minimal costs and try to implement it.

Appendix 9.2.1.5. New Alternatives Introduced Upon Failure

Failure of alternative 0 with \( \text{coding}_A = \text{coding}_B \) is a bearer control setup problem (no specialized resources in the call configuration), and therefore no new alternatives are introduced\(^\text{19}\). Upon failure of alternative \( I_{(0,b)} = (i_b,p_i) \), with \( \text{coding}_A \neq \text{coding}_B \) take the following alternatives:

1. If \( i_1 = 1 \)
   \[
   I_{(0,b)} = (\min(c,C_i(\text{coding}_A, \text{coding}_B)), N(p_i))
   \]
   \[
   I_{(0,b+1)} = (i_1+1, p_i)
   \]
   The first alternative introduces the best conversion possible at the next best node available.
   The second, newly introduced alternative, checks the next best conversion at the current node.

2. Else
   \[
   I_{(0,b)} = (i_1+1, p_i)
   \]
   Upon failure of another alternative, say \( I_{(a,b)} = (i_1, p_1, j_1, i_2, p_2, j_2, ..., j(n-1), i_n, p_n) \), where the converter \( C_{i_2}(J_{j_1}, J_{j_2}) \) is not available, introduce:

   1. if \( i_2 = 1 \) and \( n(p_2) = 1 \)
      \[
      I_{(a,b)} = (i_1, p_1, j_1, i_2+1, p_2, j_2, ..., j(n-1), i_n, p_n)
      \]
      \[
      I_{(a,b+n)} = (i_1, p_1, j_1, \min(c,C_i(J_{j_1}, J_{j_2})), N(p_2), j_2, ..., j(n-1), i_n, p_n)
      \]
      \[
      \forall (j_1(b(J_{j_1}) \geq b(J_{j_2})) \text{ and } (j_1 \neq j_2) \text{ and } (j_1 \neq j_1))
      \]
      \[
      I_{(a+n+1,1)} = (i_1, p_1, j_1, \min(c,C_i(J_{j_1}, J_{j_2})), p_2, j_2, \min(c,C_i(J_{j_1}, J_{j_2})), p_2a, j_2, ..., j(n-1), i_n, p_n)
      \]

\(^{18}\) The reduction in cost is only possible for \( \text{Cost}(%\text{Bandwidth}(i_1)) \), as \( \text{Cost}(%\text{Resources}(i_1)) = R(C_i(\text{A}, J_i)) + R(C_i(\text{J}, B)) \geq \text{Cost}(%\text{Resources}(i_1)) = R(C_i(\text{A}, B)). \)

\(^{19}\) It is possible in these cases to consider resource control trying alternative configurations (thus introducing conversions), in order to find a connection between the two parties. This decision is not part of the algorithm, but part of the RC functionality as a whole. If possible, it simply implies that RC introduces new alternatives (with conversions) and tries to implement these.

\(^{20}\) Implying it is the initial alternative.
Where \( p_{2a} \) can be defined depending on the bandwidth of the coding introduced (in general defining the node following \( P(p_2) \)).

2. if \( i_2 = 1 \)
\[
I_{(a,b)} = (i_1, p_1, j_1, i_2+1, p_2, j_2, \ldots, j(n-1), in, p_n)
I_{(a,b,max(a) + 1)} = (i_1, p_1, j_1, \min(c, C_2(J_{j_1}, J_{j_2})), N(p_2), j_2, \ldots, j(n-1), in, p_n)
\]
3. Else
\[
I_{(a,b)} = (i_1, p_1, j_1, i_2+1, p_2, j_2, \ldots, j(n-1), in, p_n)
\]

### Appendix 9.2.1.6. Proof of the Algorithm

[1] The algorithm finds the optimal configuration

Proof [1]: This statement will be proved by proving the following two statements:
[1a] The cost of each newly introduced alternative is larger than the cost of a previously introduced alternative already introduced, and
[1b] All alternative configurations are possible with this algorithm.

Proof [1a]:
Each new alternative is acquired from a previous alternative in one of three ways. For each of these new alternatives it must be proved that the alternative is more expensive in terms of total connection costs and resources used costs than at least one alternative already evaluated:

- increasing \( i_i \):
\[
(..., i_i, p_i, \ldots) < (... , i_i + 1, p_i, \ldots)
\]
This follows from the definition of \( C(A, B) \)
- increasing \( p_i \):
\[
(..., i_i, p_i, \ldots) < (... , \min(c, C_2(J_{j_a}, J_{j_b})), N(p_i), \ldots)
\]
This follows from the definition of \( p_i, N(p_i) \) and the fact that \( \min(c, C_2(J_{j_a}, J_{j_b})) \geq i_i \), because \( i_i = 1 \) in the cases where this alternative is introduced.
- introducing a new coding translation in between the existing codings, where the allocation of the specified converter fails (see Figure 9.1).
Suppose alternative (1) fails on the conversion $C(I, J, K)$. One new introduced alternative is (2).

$\text{Cost}(1) = \ldots + R(C_I(I, J, K)) + R(C_d(J, K)) + d_3 b(J_3) + \ldots$

$\text{Cost}(2) = \ldots + R(C_b(J, J_4)) + R(C_d(J_4, J_3)) + R(C_d(J_3, J_2)) + d_1 b(J_4) + d_2 b(J_3) + \ldots$

Because of definition $R(C_I(I, J, K)) \leq R(C_b(I, J_4)) + R(C_d(J_4, J_3))$ and $d_3 \leq d_1 + d_2$, thus $\text{Cost}(1) \leq \text{Cost}(2)$.

Q.E.D.

Proof [1b]:
Each possible configuration can be seen as a cascade of (a possibly infinite number of\(^{21}\)) converters, distributed in all possible ways over the network nodes. With the algorithm depicted here, each possible combination of converters with all possible locations of the converters can be created in the progress of the evaluation of alternatives.

Q.E.D.

Question remains what happens if a converter overlaps another, already introduced, converter in the cascade when it is ‘moved’ in the configuration. It is easy to prove that no new alternatives are introduced in this manner.


Proof [2]:
For each step in the algorithm, the shortest route between the two parties in the service must be recalculated, following the converters selected for the configuration and depending on the

\(^{21}\) Limited only by the number of converters present in the network (domain).
weights on the links (which, in turn, are dependent on the successive conversions realized by
the converters). The shortest route can be found using P-hard algorithms \(O(M)\), with \(M\) the
number of nodes in the network).
For the first step \(R-1\) alternatives are introduced, with \(R\) the total number of different codings.
For each failure, at most \(R-1\) new alternatives are introduced and the failed alternative yields
two new alternatives, alternatives 1 and 2. In this way, after \(f\) failures, no more than \(R(f+1)-1\)
alternatives are to be evaluated. The algorithm is therefore \(O(MR)\).

Q.E.D.

Note: The algorithm is worked out for failures from the ‘left side’ of the implicit
connection. Errors on the other side work in mirrored fashion. The complete
algorithm for an implicit connection can therefore be constructed by duplicating
the algorithm, as given here.

Appendix 9.2.1.7. Formal Notation of the Algorithm

Note: Cursive text stands for labels and programmer information.

i_Connection \((\text{ICON}, \text{A}, \text{B}, \text{Coding}_A, \text{Coding}_B, \text{Network})\)

begin

*initialize alternatives

if coding_A = coding_B then
    \(I_{(0,1)} = 0\)
else
    \(I_{(0,1)} = (1,1)\)
fi

do for all \(i\) with \((i,\text{neq.coding}_A)\) and \((J,\text{neq.coding}_B)\)
    \(I_{(i,j)} = (1,1,1,1,1)\)

od

*algorithm

\((\text{optimum.1}, \text{optimum.2}) = ((i, j))(\text{MIN}(\text{Cost}(I_{i,j})))\)

Trytoimplement\((I_{(\text{optimum.1}, \text{optimum.2})})\)

*alternative options

if Failure\((I_{(\text{optimum.1}, \text{optimum.2})})\) then
    \(i = \text{optimum.1}\)
    \(j = \text{optimum.2}\)
    \(row = I_{(i,j)}\)
if \(i=0\) then
    if (coding_A.eq.coding_B) then
        RemoveAlternative\((I_{(i,j)})\)
    else if (\((\text{row.1}),\text{eq.1})\) then
        \(I_{(i,j)} = (\text{row.1} + 1, \text{row.2})\)
        \(I_{(\text{max} (i+1))} = (\text{MIN}(c, C_i(\text{coding}_A, \text{coding}_B)), \text{N}(\text{row.2}))\)
    end
end

*N(p) calculates the next best solution after \(p.\)
else
    \( I_{i,j} = (\text{row}.1 + 1, \text{row}.2) \)
fi

else
    if \( ((\text{row}.\text{error}).eq.1) \land (n(\text{row}.(\text{error} + 1))).eq.1) \) then
        \( I_{i,j} = \text{row}.(\text{row}.\text{error} = \text{row}.\text{error} + 1) \)
        \( I_{i,\text{max}(i)+1} = \text{row}.(\text{row}.\text{error} = \text{min}(c, C_c(J_{\text{row}.(\text{error} - 1)}, J_{\text{row}.(\text{error} + 2)})); \text{row}.(\text{error} + 1) = \text{N}(\text{row}.(\text{error} + 1))) \)
        do for all \( b \) with \( ((b \neq \text{row}.(\text{error} - 1)) \land (b \neq \text{row}.(\text{error} + 2)) \) and \( (b(J_b). \leq b(J_{\text{row}.(\text{error} + 2)})) \)
            \( I_{i,\text{max}(i) + 1} = \text{row}.\text{insert}(\text{row}.\text{error} = \text{min}(c, C_c(J_{\text{row}.(\text{error} - 1)}, J_b)), \text{row}.(\text{error} + 1) = \text{row}.(\text{error} + 1), \text{row}.(\text{error} + 2) = b); \text{row}.(\text{error} + 3) = \text{min}(c, C_c(J_b, J_{\text{row}.(\text{error} + 2)})) \)
        od
    else
        \( I_{i,j} = \text{row}.(\text{row}.\text{error} = \text{row}.\text{error} + 1) \)
    fi
fi

goto *algorithm

end.

Appendix 9.2.2. Point-to-Multipoint Connection: Implicit Distribution

Appendix 9.2.2.1. Complexity of the Problem

In [CHO91] it is shown that the resource allocation problem for point-to-multipoint connections are NP-hard, not considering all possible conversions in between. For \( N \) endnodes, each possible structure can be designed by backwards combining the \( N \) endnodes into at most \( N-1 \) groups and repeating this progress recursively until all groups are combined. Each step introduces
\[
\forall(<k_i >1) \left( \sum (k_i) = N-1 \right) : \frac{(N-1)!}{\prod(k_i !)}
\]
posibilities. Looking in the other direction, all possible configurations for a \( I \) to \( N \) distribution are depicted. Secondly, all types of conversions have to be introduced for all distributors and the interconnections between the originating node, the distributors and the endnodes. The final problem is to optimally locate the necessary resources.

In this Appendix an algorithm is given, with which all alternatives, possible within the network, are compared. With every step in the algorithm, the minimum of the maximum cost for the alternatives in consideration is updated, so that alternatives that are more expensive than at least one alternative already found are not investigated.

Appendix 9.2.2.2. Syntax
### Distributed Resource Control for B-ISDN Release 2/3

#### Reference

- Ingoing Endpoint Reference
- Outgoing Endpoint References
- Coding of Ingoing Endpoint
- Coding of Outgoing Endpoints
- Description of the Network (layout)

### Definitions

- **I(A)**: Unique identifier of a configuration of all connections to endnodes passing through node A
- **A_<j>**: Unique identifier of a node in a configuration
- **A_<j>_j**: Identifier, uniquely identifying neighbouring node j of node A_<j>
- **B(I_i(A_<j>_j), A_<j>_j)**: Set of endnodes directed through node A_<j>_j in configuration alternative I_i for the set of endnodes passing through A_<j>_j
- **b_{min}**: The minimum bandwidth of any connection
- **b_{min}(<C>_)**: The minimum cost for bandwidth over the range of codings <C>
- **MST(<A>_)**: The minimum spanning tree for the network, constructed with the nodes A the branches between nodes X_1 and X_2 being C_t(coding_X_1, coding_X_2)

### Appendix 9.2.2.3. Introduce Initial Alternative Configurations

Start at the originating endnode. For each of the N destination parties, the signal can be sent in the direction of k neighbouring nodes, yielding k^N different possibilities. If k ≥ N the alternatives, in which each user is sent in a separate direction (in total \( \frac{k!}{(N-k)!} \) possibilities), do not count, as these alternatives will cost at least as much as the alternative, in which the shortest route algorithm is used for each party. This is the final alternative, introduced as an absolute maximum threshold for necessary costs. For each possible alternative the different codings must be taken into account. After introduction of the different possible codings the absolute maximum and absolute minimum costs can be calculated. All possible configurations, with absolute minimum costs less than the minimum of the absolute maximum costs are taken into account. See Figure 9.2 for a visual explanation.
For each step the absolute maximum costs for any configuration can be updated. This process is repeated until all possibilities are checked. For each step only the nodes which have more than one addressed endpoint have to be evaluated further in the algorithm (the other ones, of course have only one destination endpoint left over and the implicit connection algorithm can be used):

\[ \forall j \left( N(B(I_d(A_{<j>}, A_{<j>}) > 1)) \right). \]

**Appendix 9.2.2.4. Maximum Cost Threshold**

An easy method for calculating the maximum costs is to calculate the costs for making the shortest routes to each endnode from the node it is routed through. This should be evaluated for each possible coding of the current node and can thus be optimized (minimized).

\[
\text{MIN}\left( \forall j A_{<j>}, \forall_j A_{<j>}, \text{out} \right) \left( \text{Cost} \left( \sum_j (d(P(A_{<j>}), P(A_{<j>})))(\text{coding}_A_{<j>}, \text{out}) \right) \right)
\]

\[ \left| \left( N(B(I_d(A_{<j>}, A_{<j>}) \neq 0)) \right) + \sum_j \sum_k (i, \text{Connection}(A_{<j>}, B_k, \text{coding}_A_{<j>}, \text{out}, \text{coding}_B_k, \text{Network}))(B_k \in B(I_d(A_{<j>}), A_{<j>})) + \sum_j \sum_k \text{R}(C_e(A_{<j>}, A_{<j>}, \text{out})) \right) \]

\[ \text{Deltacost}_{\text{max}, j}(I_d(A_{<j>})) = \sum_j \sum_k (i, \text{Connection}(A_{<j>}, B_k, \text{coding}_A_{<j>}, \text{out}, \text{coding}_B_k, \text{Network}))(B_k \in B(I_d(A_{<j>}), A_{<j>}), (N(B(I_d(A_{<j>}), A_{<j>}) = 1) \land (B_k \in B(I_d(A_{<j>}), A_{<j>})))) + \sum_j \sum_k \text{R}(C_e(A_{<j>}, A_{<j>}, \text{out})) \]

\[ \text{Deltacost}_{\text{max}, \text{all}}(I_d(A_{<j>})) = \sum_j \sum_k (i, \text{Connection}(A_{<j>}, B_k, \text{coding}_A_{<j>}, \text{out}, \text{coding}_B_k, \text{Network}))(B_k \in B(I_d(A_{<j>}), A_{<j>}, (N(B(I_d(A_{<j>}), A_{<j>}) > 1)))) \]
The maximum total cost of a valid configuration can be acquired by checking all possible combined configurations.

**Appendix 9.2.2.5. Minimum Cost Threshold**

The absolute minimum cost can be found by calculating the minimal costs for conversion of the original signal to all required signals, added with the cost for transporting the signal with minimum bandwidth over the minimum total length of all connections.

\[
\text{Deltacost}_{\text{min},j}(I_i(A_{<j})) = \sum_j \left( (d(P(A_{<j})), P(A_{<j}))b_{\text{min}} (B(I_i(A_{<j})), A_{<j}) \neq 0) + \sum_j (d(P(A_{<j}), P(B_k))b_{\text{min}} (B_k \in B(I_i(A_{<j})), A_{<j}) \text{and} (N(B(I_i(A_{<j})), A_{<j}) = 1) \right) + \sum_j (d(P(A_{<j}), P(B_k))b_{\text{min}} (B_k \in B(I_i(A_{<j})), A_{<j}) \text{and} (N(B(I_i(A_{<j})), A_{<j}) > 1))
\]

The minimum total cost of a valid configuration can be acquired by checking all possible combined configurations (thus combining all parties in the call as they may have been split into subgroups, for which each different possible configuration is checked), increased by the cost for the minimum spanning tree for the conversions \(\text{MST}(A, B_k), k \in \{1, ..., N\}\).

**Appendix 9.2.2.6. Algorithm**

Take the alternative with the minimal costs and try to implement it.

**Appendix 9.2.2.7. New Alternatives Introduced upon Failure**

If a distributor fails (that is, for example a 3-output distributor (which is actually a three-way multicast) is not possible), step back into the algorithm to the point where this distributor was chosen, continue the algorithm but rule out any alternative which requires a \((3+)\)-output distributor at the specified node.

If the reservation of a converter, necessary in the configuration, fails, step back to the point where the alternative was introduced, take the second best alternative introducing a new conversion, and finish the algorithm. For the failure of an implicit connection, the alternatives, as presented in the implicit connection algorithm, can be used. Note that with the converters, mentioned before the implicit connection, converters at nodes where a distributor is located, is used.

**Appendix 9.2.2.8. Proof of the Algorithm**

[3] A finite number of alternatives has to be evaluated.

Proof [3]:
The absolute maximum cost, as calculated in the first step, equals the summation of the cost of the implicit connections to all endnodes in the distribution. For every alternative in the algorithm, the maximum cost threshold is calculated by adding the costs for implicit connections to each of the receiving endnodes in the distribution to the costs already made.
and therefore all alternatives, for which the summation of the bandwidth multiplied with the
distance to all receiving nodes (including the summation of the distances of the nodes,
reached in the alternative, to the originating node) is larger than this maximum, are no valid
alternatives and the algorithm definitely stops when it reaches these nodes$^{22}$.  

Q.E.D.

[4] This algorithm finds the optium configuration of the i_Distributor

Proof [4]:
Without the limitations set by the maximum and minimum cost criteria, all possible
alternatives will be evaluated. The configuration for the minimum cost will only in very rare
cases be the actual physical implementable implementanation yielding minimal cost, so it
certainly does not rule out any of the possible configurations. The alternative for the
maximum cost is an always implementable alternative$^{23}$, and therefore a valid maximum.

Q.E.D.


Proof [5]:  
For each step $(k^N + 1)$ alternative routings$^{24}$ of the receiving parties to the neighbouring nodes
are evaluated. For each alternative a maximum and minimum cost must be calculated. The
minimum cost can be calculated with a simple algorithm, with a complexity of $O(N + R)$,
with $R$ the number of different codings possible. The maximum cost algorithm is more
complex, as each possible coding structures of all the new introduced nodes have to be
evaluated, yielding at most $R^k$ possibilities. With this, only the implicit connections between the
nodes have to be made. The total algorithm yields a complexity of $O((k^N + 1)(R + N) R^k)$ per
node with more than one destination endnode routed through per step. This can be calculated
for each node and in this manner the total complexity can be obtained.

Q.E.D.

Note: The initial maximum is set by setting up point-to-point connections with each of
the receiving parties. If all less expensive alternatives fail and this alternative fails
as well, the algorithm ends without finding a possible solution. This exceptional
case can be circumvented by increasing CMAX if the algorithm stops without
finding a solution$^{25}$.

Appendix 9.2.2.9. Formal Notation of the Algorithm

\[ \text{IDistributor} \quad (\text{IDistr, A, B}_1 \ldots \text{B}_N, \text{Coding}_A, \text{Coding}_B_1 \ldots \text{B}_N, \text{Network}) \]

$^{22}$ The maximum threshold is updated for every step in the algorithm and can never grow (not taking
into account situations where desired resources are not available), so the maximum set found the first
step of the algorithm is definitely an valid upper bound.

$^{23}$ If the required resources are available.

$^{24} k^N - \frac{k!}{(k-N)!} + 1$ if $k \geq N$.

$^{25}$ A possibility is to predefine a maximum with a threshold value.
begin

*initialize alternatives

C\(\text{MAX} = \sum_{A} \text{i:\text{Connection}(A, B, \text{coding}_A, \text{coding}_B, \text{Network})}\)

\text{Configurations}(0, A, <B>, \text{MST}(A, <B>), 0, \text{CMAX})

*minimize costs for alternatives

do for all \(I_{1}(<I>)\)

\text{MinimizeCosts}(I_{1})

od

*algorithm

optimum = (i\text{MIN}(\text{Cost}(I_{1})))

\text{TryToImplement}(\text{optimum})

*alternative options

if Failure(optimum) then

case Failure of

\text{StepBackToDistributor}

\text{StepBackToConverter}

\text{StepBackToAlternative}

end

\text{RemoveAlternative}

\text{Configurations}(<I> \text{remaining}, <(A(I)I_{1} \in <I> \text{remaining})>, <(B(I)I_{1} \in <I> \text{remaining})>, <\text{Cost}_{\text{min},t,f}(I_{1}I_{1} \in <I> \text{remaining})>, <\text{Cost}_{\text{max},t,f}(I_{1}I_{1} \in <I> \text{remaining})>, \text{CMAX remaining})

goto *minimize costs for alternatives

fi

end.

\text{Configurations}(<I>, <(A(I)I_{1} \in <I>)>, <(B(I), A(I)I_{1} \in <I>)>, <\text{Cost}_{\text{min},t,f}(I_{1}I_{1} \in <I>)>, <\text{Cost}_{\text{max},t,f}(I_{1}I_{1} \in <I>)>, \text{CMAX})

<\text{new}_{I} = \{\}

<\text{new}_{A(I)I_{1} \in <I>} = \{\}

<\text{new}_{B(I), A(I)I_{1} \in <I>} = \{\}

<\text{new}_{\text{Cost}_{\text{min},t,f}(I_{1}I_{1} \in <I})} = \{\}

<\text{new}_{\text{Cost}_{\text{max},t,f}(I_{1}I_{1} \in <I})} = \{\}

begin

*calculate minimum and maximum costs for all possible configurations

\text{do for all } I_{1}(<I>)

\text{do for all } A_{j}(<A(I)I_{1} \in <I>)

define all configurations \(I_{j}(A_{j})\)

\text{Calc}(\text{Deltacost}_{\text{min},t}(I_{j}(A_{j})))

\text{Calc}(\text{Deltacost}_{\text{max},t}(I_{j}(A_{j})))

end.
**Appendix 9.2.3. Multipoint-to-Point Connection: Implicit Combination**

**Appendix 9.2.3.1. Complexity of the Problem**

The problem for the optimal configuration of an implicit combination is identical to the configuration problem of an implicit distribution. The only exception is that for implicit combiners the costs for combiners have to be taken into account, whereas the costs for distributors (multicasting) is 0. This because multicasting requires no special resources.
Appendix 9.2.3.2. Syntax

i_Combiner

( 
IComb,
A₁ ... Aₙ,
B,
Coding_A₁ ... Aₙ,
Coding_B,
Algo,
Network
)

The introduction of the costs for combiners introduces the following changes in the algorithm for the configuration of an implicit distributor:

Appendix 9.2.3.3. Maximum Cost Threshold

An easy method for calculating the maximum costs is to calculate the costs for making the shortest routes for each endnode from the node it is routed through. This should be evaluated for each possible coding of the current node.

\[
\text{MIN}( \bigwedge_j A_{<j>in}, \bigwedge_j A_{<j>out} \ (\text{Cost}(\Sigma_j (d(P(A_{<j>}), P(A_{<j>}) \text{(coding}_A_{<j>out}) \\
\text{I}(N(B(I_{<j>}(A_{<j>}), A_{<j>}) \neq 0)))) + \Sigma_j (i_{\text{Connection}}(I_{<j>}, B_k, \text{coding}_A_{<j>out}, \\
\text{coding}_B_k, \text{Network}))(B_k \in B(I_{<j>}(A_{<j>}), A_{<j>})) + \Sigma_j R(C_v(A_{<j>out}, A_{<j>out})) + \\
R((N_a C_b + N_b C_v)(b(\text{coding}_A_{<j>out})) \text{I}(N(B(I_{<j>}, A_{<j>}) \neq 0))>1))))
\]

\[
\Delta_{\text{costmax}}(I(A_{<j>})) = \Sigma_j (d(P(A_{<j>}), P(A_{<j>}) \text{(coding}_A_{<j>out}) \text{I}(N(B(I_{<j>}(A_{<j>}), A_{<j>}) \neq 0)))) + \\
\Sigma_k (i_{\text{Connection}}(A_{<j>}, B_k, \text{coding}_A_{<j>out}, \text{coding}_B_k, \\
\text{Network}))(B_k \in B(I_{<j>}(A_{<j>}), A_{<j>})) + \Sigma_j R(C_v(A_{<j>out}, A_{<j>out})) + R((C_b + N C_v)(b(\text{coding}_A_{<j>out})) \text{I}(N(B(I_{<j>}, A_{<j>}) \neq 0))>1))
\]

\[
\Delta_{\text{costmax}}(I(A_{<j>})) = \Sigma_j \Sigma_k (i_{\text{Connection}}(A_{<j>}, B_k, \text{coding}_A_{<j>out}, \text{coding}_B_k, \\
\text{Network}))(B_k \in B(I_{<j>}(A_{<j>}), A_{<j>}) \text{and}(N(B(I_{<j>}(A_{<j>}), A_{<j>})) > 1)))) + \Sigma_j R(N_{ak} C_b + \\
N_{bk} C_v)(b(\text{coding}_A_{<j>out}))
\]

with

\[
N_a = \begin{cases} 
0 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) > 1 \\
1 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) \leq 1
\end{cases}
\]

\[
N_b = \begin{cases} 
0 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) > 1 \\
1 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) \leq 1
\end{cases}
\]

\[
N_{ak} = \begin{cases} 
0 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) > 1 \\
1 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) \leq 1
\end{cases}
\]

\[
N_{bk} = \begin{cases} 
0 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) > 1 \\
1 & \text{I}(N(B(I_{<j>}, A_{<j>})) \neq 0) \leq 1
\end{cases}
\]
Note: an $N$-input combiner can be constructed with multiple combiner at a single node. This would increase the costs for the alternative, and $N_a$ and $N_b$ would change accordingly. These alternatives can be introduced upon failed reservation of a combiner.

The maximum total cost of a valid configuration can be acquired by checking all possible combined configurations.

**Appendix 9.2.3.4. Minimum Cost Threshold**

The absolute minimum cost can be found by calculating the minimal costs for conversion of the original signal to all required signals, added with the cost for transporting the signal with minimum bandwidth over the minimum total length of all connections.

$$\text{Deltacost}_{\text{min}}(I_i(A_i\searrow)) = \sum_j ((d(P(A_i\searrow), P(A_i\searrow))b_{\text{min}})(B(I_i(A_i\searrow)), A_i\searrow) + (B(I_i(A_i\searrow), A_i\searrow) \neq 0) + R(N_aC_b + N_bC_v)b_{\text{min}})$$

The minimum total cost of a valid configuration can be acquired by checking all possible combined configurations, increased by the cost for the minimum spanning tree for the conversions $\text{MST}(A, <B_k>)$.

Note: As well as for the maximum cost threshold it must be taken into account that an $N$-input combiner can be constructed with multiple combiner at a single node. This would increase the costs for the alternative, and $N_a$ and $N_b$ would change accordingly. These alternatives can be introduced upon failed reservation of a combiner.

**Appendix 9.2.3.5. Formal Notation of the Algorithm**

**ICombiner**

**(IComb, A_1, ..., A_N, B, Coding_A_1, ..., Coding_A_N, Coding_B, Algo, Network)**

begin

*initialize alternatives

$\text{CMAX} = \sum_k (\text{i\_Connection}(A_k, B_k, \text{coding}_A, \text{coding}_B, \text{Network}) + R(N_aC_b + N_bC_v)b_{\text{min}})$

**Configurations**(0, A, <B_k>, 0, 0, CMAX)

*minimize costs for alternatives

do for all $I_i(<i>)$

MinimizeCosts($I_i$)

od

*algorithm
optimum = (iMIN(Cost(I)))
TryToImplement(1_{optimum})

*alternative options

if Failure(1_{optimum}) then
    case Failure of
        combiner
            StepBackToCombiner
        converter
            StepBackToConverter
        i_connection
            StepBackToAlternative
    end
    RemoveAlternative
    Configurations(1_{remaining}, <(A(I), I_{remaining}>>) remaining, <(B(I), A(I), I_{remaining}>>) remaining, <Cost_{min},(I), I_{remaining}>>, <Cost_{max},(I), I_{remaining}>>, C_{MAX} remaining)
goto *minimize costs for alternatives
end.

Configuration(1, <(A(I), I_{remaining}>>) remaining, <(B(I), A(I), I_{remaining}>>) remaining, <Cost_{min},(I), I_{remaining}>>, <Cost_{max},(I), I_{remaining}>>, C_{MAX} remaining)

1_{new} = {}
A(III) in 1_{new} = {}
B(I, A(I)) in 1_{new} = {}
COSTMIN(III) in 1_{new} = {}
COSTMAX(III) in 1_{new} = {}

begin
    *calculate minimum and maximum costs for all possible configurations
    do for all I (1) do for all A_{2} <(A(I), I_{remaining}>>) define all configurations I_{i}(A_{2})
        Calc(Deltacost_{min},(I_{i}(A_{2})))
        Calc(Deltacost_{max},(I_{i}(A_{2})))
    od
    *calculate maximum and minimum costs for all combinations of configurations
    do for all <I> (<(I)>>) *strings of combinations
        Cost_{min},(I) = \Sigma Deltacost_{min}(I_{i}(A_{2})) + Cost_{min},(I)
        Cost_{min},(I) = \Sigma Deltacost_{min}(I_{i}(A_{2})) + Cost_{min},(I)
        Cost_{max},(I) = \Sigma Deltacost_{max}(I_{i}(A_{2})) + Cost_{max},(I)
        Cost_{max},(I) = \Sigma Deltacost_{max}(I_{i}(A_{2})) + Cost_{max},(I)
        C_{MAX} = MIN(C_{MAX}, C_{max},(I))
    od
    * continue with all possible combinations of configurations
do for all \(<i_{ij}\rangle \ (<i_{ij}\rangle)\) *strings of combinations
   if \((\text{Cost}_{\text{min}}(\langle i_{ij}\rangle) < \text{CMAX})\) then
      \(<i_{new} = \langle i_{new}, \langle i_{ij}\rangle\rangle\)\)
      \(<A(i_{III} \in \langle i_{new}\rangle)\rangle_{new} = \{\langle A(i_{III} \in \langle i_{new}\rangle)\rangle_{new}, \langle A(i_{ij} \in \langle i_{ij}\rangle)\rangle\}\)
      \(<B(i, A(i))\in \langle i_{new}\rangle)\rangle_{new} = \{\langle B(i, A(i))\in \langle i_{new}\rangle)\rangle_{new}, \langle B(i_{ij}, A(i_{ij})\in \langle i_{ij}\rangle)\rangle\}\)
      \(<\text{COST}_{\text{MIN}}(\langle i_{III} \in \langle i_{new}\rangle)\rangle_{new} = \{\langle \text{COST}_{\text{MIN}}(\langle i_{III} \in \langle i_{new}\rangle)\rangle_{new}, \langle \text{Cost}_{\text{min},t,tC}(i_{ij})\rangle\}\)
      \(<\text{COST}_{\text{MAX}}(\langle i_{II} \in \langle i_{new}\rangle)\rangle_{new} = \{\langle \text{COST}_{\text{MAX}}(\langle i_{II} \in \langle i_{new}\rangle)\rangle_{new}, \langle \text{Cost}_{\text{max},t,tC}(i_{ij})\rangle\}\)
   fi
od

\textbf{Configurations}(\langle i_{new}\rangle, \langle A(i_{ij})\in \langle i_{ij}\rangle)\rangle_{new}, \langle B(i_{ij}, A(i_{ij})\in \langle i_{ij}\rangle)\rangle_{new}, \langle \text{COST}_{\text{MIN}}(\langle i_{III} \in \langle i_{new}\rangle)\rangle_{new}, \langle \text{COST}_{\text{MAX}}(\langle i_{II} \in \langle i_{new}\rangle)\rangle_{new}, \langle \text{CMAX}\rangle\)

end.

\textbf{MinimizeCosts}(\langle i_{ij}\rangle)

begin
\text{MIN}(\Sigma_{\langle i\rangle}(\langle A_{\langle i\rangle}, A_{\langle i\rangle}\rangle), \Sigma_{\langle i\rangle} (\langle i\rangle_{\text{Connection}}(A_{\langle i\rangle}, A_{\langle i\rangle\prime}), A_{\langle i\rangle\prime}, A_{\langle i\rangle\prime}, \text{Network}))(N(B(i, A_{\langle i\rangle}), \langle i\rangle > 1)) + \Sigma_{\langle i\rangle} (\langle i\rangle_{\text{Connection}}(A_{\langle i\rangle}, B_{\prime}, A_{\langle i\rangle\prime}, B_{\prime}, \text{Network}))(B_{\prime} \in B(i, A_{\langle i\rangle}), \langle i\rangle_{\text{Connection}}(A_{\langle i\rangle\prime}, A_{\langle i\rangle\prime}, \text{Network}))(N(B(i, A_{\langle i\rangle\prime}), A_{\langle i\rangle\prime} > 1)) + R(C_{\langle i\rangle}(A_{\langle i\rangle}, A_{\langle i\rangle\prime}, \text{Coding}_{\langle i\rangle}, A_{\langle i\rangle\prime}, \text{Coding}_{\langle i\rangle\prime})) + R((C_{\prime} + N C_{\prime})(b(\text{coding}_{\langle i\rangle}, \langle i\rangle)))

end.

Appendix 9.2.3.6. Additional Information

The method is valid for both component and composit mapping. For component mapping the full bandwidth must be used for each connection, for the composit model each connection (and resource) can be optimized for bandwidth. If signals to receiving parties are combined over the same trunk, the bandwidth necessary is equal to the maximum bandwidth of any of the receiving parties:

\[ b = \langle \text{MAX}(b, (t = \text{receiving party}))\rangle. \]

Appendix 9.2.4. Multipoint-to-Multipoint Connection: Implicit Bridging (Component Mapping)

Appendix 9.2.4.1. Complexity of the Problem

For easy description it is considered here that each party has access to the full bandwidth signal of each other party involved in the abstract service. If this is not the case, the alternative configurations can be updated by removing all direct connections (i.e. not via a bridge) from any party to all parties, that do not have access to its signals (deductable from the access lists (chapter 6)). The restriction mentioned here (component downstream mapping...
for all parties) significantly diminishes the number of configuration possibilities. For $N$ parties $(N-1)$ alternatives are to be checked$^{26}$.

![Figure 9.3: Various alternative configurations for an N-input Bridge.](image)

For each of these $(N-1)$ alternatives all alternative codings for each bridge and connection have to be checked, as well as the different configurations for the combiners and distributors, taking the different endpoint codings into account. Except for the first alternative, with an $N$-input bridge, the alternatives are very complex to insert into an algorithm, as each implicit object has endpoints connected to other implicit objects. This implies that in any alternative the different configurations for all implicit objects have to be evaluated simultaneously, leading to a great number of possibilities. Therefore, no algorithm is developed which finds the optimal configuration in terms of resources used and total connection costs, using an exceptable amount of time. In section 9.3 a few restrictions are considered, with which very simple algorithms with very adequate results can be constructed.

**Appendix 9.2.4.2. Syntax**

$$i_{\text{Bridge}}(\text{IBridge, J}_{1, \text{in}}, J_{1, \text{out}}, J_{N, \text{in}}, J_{N, \text{out}}, \text{Coding}_{J_{1, \text{in}}}, ..., \text{Coding}_{J_{N, \text{out}}})$$

Reference

Endpoint references

Coding of the endpoints

---

$^{26}$ The alternatives can be obtained by taking an N-input/output bridge and 'pulling' out parties one by one, until the bridge is empty. Note that the alternative with a 1-input bridge is a non-existing one.
Appendix 9.2.5. Multipoint-to-Multipoint Connection: Implicit Bridging (Composit Mapping)

Appendix 9.2.5.1. Complexity of the Problem

It is not possible to completely describe the possible configurations of a logical model of a bridge involving composit downstream mapping to the parties involved. Each signal flow from a party to any receiving party can pass thorough an infinite number of combiners, distributers, converters and bridges, in any order, as long as the end-to-end relation of the signal remains identical. Most of these configurations are absurd, but the recursive combination and distribution of the signals may save on total connection costs. Two very important observations can be made:

- For parties that are close together and (a part of) the received (composit) signals overlap(s) the signal flows to the parties may be combined to save on total connection costs.
- For parties that are distant and (a part of) the received composit signals overlap(s) the combined signals flow may be distributed from a certain point in order to save total connection costs.

Many issues within these obeservations remain vague, however. When are parties considered close together? When is the overlapping part of the signals large enough to consider combination or combined combination and distribution? Essence of the observations is that combination of the signalflows to multiple parties can be considered if the gain on total connection costs surpasses the increased costs of resources used. However, the problem is more complex than depicted by these observatons. Additional problems are:

- The combination and distribution of signals can be repeated recursively.
- The concurred combination and distribution of a mixture of signals between (a group of) users can be implemented (in terms of specialized resources) with a bridge or bridges instead of distributers and combiners.

It is clear that the number of possible configurations for implicit bridging with composit downstream mapping is endless. It is beyond the scope of this paper to try and solve, or even describe, the problem in full. In this section, two different approaches to the problem are described.

Note: For the combination of both composit and component outputs within a single implicit bridge the algorithms for component and composit mapping can be
combined. One must bear in mind that for component downstream mapping the receiving party must have access to the full bandwidth of the signal of each party. it has access rights to (from the access list).

Appendix 9.2.5.2. Approach 1: Configuration Involving Bridging

The approach described here is actually an elaboration of the algorithm for bridging involving parties with component downstream mapping, allowing only the configuration with an \(N\)-input/output bridge. This bridge, however, can be constructed from multiple bridges. This does not optimize the result, as the total connection costs are already optimized and the resource costs increase with the introduction of each new bridge, but it may diminish setup times, when considering the case that smaller bridges (i.e. bridges with less inputs) are more frequent in the network. In Figure 9.4 an example is given.

![Figure 9.4: Splitting a bridge into multiple bridges.](image)

A second advantage is dependent on the layout of the network. If the network is divided into smaller subdomains (as is the case with the model presented in this paper), combining the signals from the parties in a single subdomain to other subdomains diminishes the number of VCI/VPI translations. This is not an advantage speaking in terms of resources used and total connection costs, but this approach is a very realistic one and these extra advantages only validate the choice for it.
As the approach only seems to make sense in case the parties involved can somehow be grouped into subgroups of parties somehow belonging together, the algorithm is worked out with the model presented in chapter 5 in mind. Note that this is a restriction to the effectiveness of the algorithm, not to the eventual functionality of it.

It is very important to stress that this algorithm does not optimize the configuration, but yields a method of finding an implementable alternative.

Appendix 9.2.5.2.1. Syntax

\[ i_{-}Bridge \] (IBridge, Reference
\[ J_{1,in}, J_{1,out}, J_{N,in}, J_{N,out}, \] Endpoint references
\[ Coding_{J_{1,in}}, ..., Coding_{J_{N,out}}, \] Coding of the endpoints
\( (\text{composit/(component + access list)}) \)
\[ \text{Algo}_1, ..., \text{Algo}_N, \] Combining algorithm for each output
\[ \text{Network} \] Layout of the Network

\[ N: \text{number of parties connected.} \]

Appendix 9.2.5.2.2. Introduce Initial Alternatives

For each local domain \( i \), a bridge with \( N_i \) inputs/outputs is recquired, with \( N_i = N(\text{parties} \in i) + (I - 1) \), where \( I \) equals the number of subdomains that also have a bridge. The coding off each bridge equals the coding of the majority of the parties connected.

The necessary bandwidth for the connections between parties and bridges and bridges inbetween can be derived from the algorithms for the outputs of the bridge.

Appendix 9.2.5.2.3. New Alternatives Introduced upon Failure

If the alternative fails on a bridge, say bridge \( B_i \), move the bridge to a local domain which yields a minimal increase in total connection costs.

Appendix 9.2.5.2.4. Formal Description of the Algorithm

\[ \text{IBridgeA} \ (\text{IBridge, } J_{1,in}, J_{1,out}, J_{N,in}, J_{N,out}, \text{Coding}_{J_{1,in}}, ..., \text{Coding}_{J_{N,out}}, \text{Algo}_1, ..., \text{Algo}_N, \text{Network}) \]

*\text{initialize alternative}*

\text{do} for all subdomains \( i \)
DefineCodingOfBridge(i)
CalcSizeOfBridge(i)

*algorithm

do for all i
TryToLocateBridge(i)

*alternative options

if Failure(i) then
MoveBridgeToSuboptimalLocation(i)
goto *algorithm
fi

do for all i
  do for A(j,j) (all parties in i)
    i_Connection(Bi, A(j,j), coding_Bi, coding_A(j,j), Network)
    i_Connection(A(j,j), Bi, coding_A(i,j), coding_Bj, Network)
  od
  do for all j (j ≠ i)
    i_Connection(Bj, Bj, coding_Bj, coding_Bj, Network)
  od
od

Appendix 9.2.5.2.5. Additional Remarks

Many options are left out here, to guarantee that the algorithm quickly finds a solution, if available. Alternative options include:

- In the algorithm a bridge is allocated at each subdomain, where parties involved in the service are located. It might be advantageous to combine the collection of parties from more than one local subdomain into one bridge, for example if for a certain subdomain only one party, involved in the service is located there.
- When introducing alternatives, the options in which the party connections of multiple subdomain overlap, it is (of course) preferable to allocate a single large bridge, combining all the signals of all the considered parties.
- Only one type of coding is considered per bridge. The Algorithm can very easily be extended to include different coding structures for each bridge.

Appendix 9.2.5.3. Approach 2: Configurations Not Involving Bridging

In [CHO95] the ORAOP algorithm is given. ORAOP is a greedy algorithm, which finds configurations optimizing the costs for resources and the total connection costs, considering several limitations. These limitations are:
• No bridges are considered.
• A preassumption on the coding structures of the different specialized resources is made and the signals of the associated users are converted accordingly.
• Only one type of conversion between two different signal coding structures is considered (no cascaded conversion).

The OROAP algorithm is an exponential algorithm ([CHO95]). In the same article, several heuristic algorithms are presented as well, that differ in how each individual resource is allocated and what the allocation order of the different resources required in a logical connection is. The algorithms are described per resource type, as mentioned in [CHO95]. For more insight into the algorithms, please refer to the article.

Appendix 9.3. Heuristic Algorithms

In this Appendix a selection of heuristic algorithms, finding configurations of service request while consuming less time, are given.

Appendix 9.3.1. Point-to-Point Connection: Implicit Connection

Appendix 9.3.1.1. Description and Validation of the Algorithm

The alternative solution does not consider cases with more than one conversion. This algorithm is therefore identical to the algorithm described in section 9.2, using only alternatives $I_{0j}$ (notated as $I_j$). Validation for this algorithm lies in the assumption that all converters for direct conversion of any coding structure to any other coding structure are available within the network.

Appendix 9.3.1.2. Introduce Initial Alternative Configurations

If $coding_A = coding_B$ then alternative $I_{(l)} = ()$. This alternative is passed on to bearer control immediately, as no alternatives are possible on RC level. If $coding_A \neq coding_B$ then alternative $I_{(l)} = (1,1) = (C_l(coding_A, coding_B), 1)$

Appendix 9.3.1.3. Algorithm

Take the alternative introducing minimal costs and try to implement it.

Appendix 9.3.1.4. New Alternatives Introduced Upon Failure

Upon failure of alternative $I_{(b)} = I(i1,p1)$, with $coding_A \neq coding_B$ take the following alternatives:

1. If $i1 = 1$
So the algorithm does allow multiple converters in the conversion of a signal. However, this cascade of converters has to be allocated at the same node.

2. Else
\[ I_{(b)} = (il + 1, p_1) \]

**Appendix 9.3.1.5. Complexity of the Algorithm**

The complexity of the algorithm is easily obtained. For each step in the algorithm, the shortest route (which depends on the bandwidth) is calculated. For the calculation of the shortest route P-hard solutions are at hand. Each step introduces at most one new alternative, so the algorithm is P-hard.

**Appendix 9.3.1.6. Limitations of the Algorithm**

The algorithm checks only on the direct availability of a signal conversion from coding_A to coding_B. If this conversion is not possible at a single node, the algorithm fails in finding a solution. However, it is considered that for all possible coding structures conversions to all other coding structures are at hand.

**Appendix 9.3.1.7. Formal Description of the Algorithm**

\[ I_{(b)} = (il,N(p_1)) \]
\[ I_{(b_{\text{max}} + 1)} = (il + 1,p_1) \]

\[ I_{(b)} = (il + 1, p_1) \]

---

Note here that the new alternative introduced is not \( \min_c(C_c(coding_A, coding_B), N(P_i)) \), as it is clear that the introduced conversion at this node can only come from this step in the algorithm.
Appendix 9.3.2. Point-to-Multipoint Connection and Multipoint-to-Point Connection: Implicit Distribution or Combination (Component Mapping)

Appendix 9.3.2.1. Description and Validation of the Algorithm

The heuristic algorithm introduced here is almost identical to the algorithm described in section 9.2. The only difference lies in the fact that the coding structure of all distributors and combiners within the configuration are identical. In this fashion the number of alternatives per step are greatly diminished. Validation follows from the supposition that, in a distribution or combination, most parties have the same endpoint coding and therefore conversion is not required. A second advantage is that it allows for distribution to be solved at BC level\textsuperscript{28}.

As the algorithm is practically identical to the algorithm described in section 9.2 it is not worked out. The only changes are the reduced number of possibilities as a result of the limitation to one coding structure and some minor changes in the maximum and minimum costs threshold.

Appendix 9.3.3. Implicit Combination (Composit Mapping)

For implicit combination of signals, with a composit signal structure, the total connection costs can be easily optimized by using the shortest route algorithm from each sending party to the receiving party. The minimum costs for the combination of signals is achieved by using a single $N$-input combiner for the combination of the signals from $N$ parties. The algorithm described here only finds a solution if an $N$-input combiner is available or two combiners are available along the shortest route paths, such that all signals are combined by the two combiners.

The coding used for the combiners is arbitrary. The most logical choice would be the coding of the majority of the connected parties, introducing the necessary costs for the converters. The total connection costs can be optimized. For the sending parties that do have a different coding structure than the combiner an implicit connections is found after the implicit combiner has been configured. Validation of this decision lies in the assumption that combiners will be less available than converters and the cost for converters.

\textsuperscript{28} And thus allowing multicasting at BCTEX's, which of course, is physically possible.
The algorithm can be easily extended to include options with different coding structures of the combiner(s).

Appendix 9.3.3.1. Introduce Initial Alternative Configurations

Calculate the shortest route from each of the sending parties to the receiving party. The optimal alternative is achieved by combining all signals with an \( N \)-input combiner at the destination node (where the receiving party is located).

\[ I_{(1)} = (\text{combiner}_N, p_1). \]

Appendix 9.3.3.2. Algorithm

Take the alternative introducing minimal costs and try to implement it.

Appendix 9.3.3.3. New Alternatives Introduced Upon Failure

Upon failure of alternative \( I_{1,j} \) introduce:

1. if initial alternative fails:
   \[ I_{(1,j+1)} = (\text{combiner}_N, N(p)) \]
   \[ I_{(2,1)} = ((\sum_{k=1}^{N} \text{combiner}_{kl}(\sum_{j=1}^{N} k = N+1)), p_1 p_2) \]

2. else:
   \[ I_{(1)} = (\text{combiner}_N, N(p)) \]

Upon failure of alternative \( I_{3} \) introduce:

\[ I_{(2)} = ((\sum_{k=1}^{N} \text{combiner}_{kl}(\sum_{j=1}^{N} k = N+1)), N(p)) \]

Appendix 9.3.3.4. Complexity of the Algorithm

The algorithm calculates the shortest routes and tries to allocate a \( N \)-input combiner at the receiving endnode in the first step. Upon each failure at most 2 new alternatives are introduced, so the algorithm is \( \text{P-hard} \).

Appendix 9.3.3.5. Formal Description of the Algorithm

\begin{verbatim}
ICombiner1 = (IComb, A, B1 ... Bn, Coding_A, Coding_B1 ... Bn, Network)

*initialize alternatives

begin

I1 = (N, 1)
I2 = 0  *dummy value

*algorithm
\end{verbatim}
optimum = (iMIN(Cost(Ij)))
TrytoImplement(1optimum)

*alternative options

if Failure(1optimum) then
row := 1optimum
if (optimum.eq.1) then
    if (I2 = 0) then
        l(2) = FindOptimalTwoConverters
    else
        l(2) = FindOptimalTwoConverters
    fi
fi
fi
end.

Appendix 9.3.3.6. Additional Information

The algorithm can be extended to include options where the locations for the combiners (in the case at least two combiners are used) are not within the tree spanned by the shortest routes. This, however, would result in complex algorithms for finding sub-optimal locations and the time gain would diminishing accordingly.

Another option is to extend the algorithm to include options with more than two combiners, but the introduction of each new combiner increases the cost for resources used with \( R(C_b + C_v) b(coding) \). However, not all options using only two converters are considered, and as these can be cheaper it is possible that the costs for the configurations rise to fast because a great deal of alternatives are not evaluated. This, however, could be desirable for calls requiring quick setup, such as e.g. emergency calls.

Appendix 9.3.4. Implicit Bridging (Component Mapping)

In the algorithm presented here the possible alternatives are restricted to the cases with N-input/output bridges. Validation for this restriction lies in the fact that:

- The costs for resources in the other alternatives compared to alternative 1 equals
  \[ \text{Deltacost} = (i - 1)R(B_c)(b(coding_{alternative_i}) - b(coding_{alternative_j})) + 2 \sum_j (C_b + N_j C_v)b(coding_{A_j}), \]
  which, in general is larger than 0. The total connection costs will definitely be as large as for alternative 1 (for each party, involved in the bridging, the minimum downstream and upstream signal bandwidth is \( b_{min} \), which is achieved only in alternative 1).

- When offering users the possibility to request services from the network involving N-input/output bridging, it is considered that the necessary resources for the simplest configuration (being an N-input/output bridge) are available.

Appendix 9.3.4.1. Introduce Initial Alternative Configurations
The alternatives involve different coding structures of the N-input/output bridge and the location of the bridge. In this section, two almost identical algorithms are depicted: The first algorithm finds the optimal bridge supposing the optimal connections to the users are implementable and consecutively configures the connections between all parties and the bridge. The second algorithm configures both the connections and the bridge at the same time. We will use the same algorithm as presented earlier in this Appendix for the implicit connections.

The initial alternatives are the same for both algorithms. However, the annotation is different as the second algorithm allows more alternatives:

**ALGORITHM IBRIDGE1:**

\[ I_{0j} = (j, p_j) \]

where \( j \) is the unique identifier of coding structure \( J_j \) of the bridge and \( p_j \) the optimal location for the bridge, thus minimizing the costs for resources used and total connection costs:

\[
\text{Cost(Resources)}(I_{0j}) = R(B_b + N B_v) b(J_j) + \sum_i R(C_i(A_i, \text{Bridge}_j))
\]

\[
\text{Cost(Connection)}(I_{0j}) = \sum_i d(P(A_i), P(\text{Bridge}_j))(\text{MIN}(b(coding_{A_i}), b(J_j)))
\]

**ALGORITHM IBRIDGE2:**

\[ I_{0,j.1} = (j, p_j, i1, p_1, i2, p_2, \ldots, iN, p_N) \]

where \( j \) is the unique identifier of coding structure \( J_j \) of the bridge and \( p_j \) the optimal location for the bridge, thus minimizing the costs for resources used and total connection costs:

\[
\text{Cost(Resources)}(I_{0,j.1}) = R(B_b + N B_v) b(J_j) + \sum_i R(C_i(A_i, \text{Bridge}_j))
\]

\[
\text{Cost(Connection)}(I_{0,j.1}) = \sum_i d(P(A_i), P(\text{Bridge}_j))(\text{MIN}(b(coding_{A_i}), b(J_j)))
\]

\( i1, p1, i2, p2, \ldots, iN, pN \) represent the optimal locations and implementations of the necessary converters for the parties, connected to the bridge. If the party has the same coding structure as the bridge, no converter is necessary and both values are 0.

**Appendix 9.3.4.2. Algorithm**

Take the alternative yielding minimal costs and try to implement it.

**Appendix 9.3.4.3. New Alternatives Introduced Upon Failure**

Here, a difference between the two algorithms is distinguishable.

**ALGORITHM IBRIDGE1:**

Upon failure of the reservation of a bridge in the alternative, the sub-optimal location of the bridge is chosen:

\[ I_{0j} = (j, N(p_j)) \]

Failed reservation of a necessary converter is handled in the algorithm for implicit connections and alternative configurations are presented according to the algorithm used for the implicit connections.

**ALGORITHM IBRIDGE2:**
Upon failure of the reservation of a bridge in the alternative, the sub-optimal location of the bridge is chosen:

$$I_{(i,j)} = (j, N(p_{j,i}), i1a, p_{1a}, i2a, p_{2a}, ..., iNa, p_{Na})$$

so that $ixa$ and $p_{xa}$ are optimal.

Upon the failed reservation of a converter, say $ij$ at $P(p_{ij})$, the same procedure is followed as in algorithm 1Connection1. However, if the attempt to reserve the necessary resources was the first with the specified bridge at that location (i.e. $ij$ and $p_{ij}$ are optimal), the following alternative is introduced as well:

$$I_{(i,j+1)} = (j, N(p_{j,i}), i1a, p_{1a}, i2a, p_{2a}, ..., iNa, p_{Na})$$

so that $ixa$ and $p_{xa}$ are optimal.

### Appendix 9.3.4.4. Complexity of the Algorithm

Two algorithms are presented here, which differ only slightly but may lead to considerable differences in time. The first algorithm optimizes only the location and coding structure of the bridge, supposing the necessary converters are available at the optimal locations. The second algorithm optimizes both the converters and the bridge at the same time.

The optimal location of the bridge can be found using a minimum sum-distance algorithm. If the distances used in the calculations represent topographic distances, the sum-distance is a convex function and simple algorithms are at hand. If this is not the case, all nodes have to be checked. The algorithm is therefore $O(M)$, with $M$ the number of nodes in the network.

For the implicit connections the algorithm 1Connection1 is used, which is a simple P-hard algorithm and therefore both algorithms presented here are P-hard.

### Appendix 9.3.4.5. Limitations of the algorithm

By limiting the alternatives in the algorithm to those involving an N-input/output bridge, the algorithm has become very easy and should quickly find a valid solution. Whether the optimal configuration can be found depends on the costs for the resources in the other alternatives (which are not considered as valid alternatives in this algorithm) and the relative gain in total connection costs achievable by using a single N-input/output bridge.

### Appendix 9.3.4.6. Formal Description of the Algorithm 1: First allocating the bridge, then optimizing the implicit connections

#### IBridge1

(IBridge, Component, $J_{1, in}, J_{1, out}, ..., J_{N, in}, J_{N, out}$, Coding$_{J_{1, in}},$ Coding$_{J_{N, out}},$ Algo$_{J_{1}}, ..., $ Algo$_{J_{N}},$ Network)

**begin**

*Initialize alternatives

**do** for all configurations $I_{(i)}$ **row** $I_{(i,j)} = (j, p)$ representing all coding structures and locations

---

29 which is not necessarily the case if the operator wishes to control traffic flows by decreasing/increasing the distance penalties.
FindOptimalLocation(I_{0})

*algorithm

optimum = (j(MIN(Cost(I_{0})))

TryToLocateBridge(I_{optimum})

*alternative options

if Failure(I_{optimum}) then

row = I_{optimum}

row.2 = N(row.2)  \text{ \text{*N(p) calculates the next best solution after p.}}

goto *algorithm

fi

do for all i

i\_Connection(A_i, B_j, coding_{A_i, j}, Network)

do

FindOptimalLocation(I_{0})

begin

Cost(Resources(I_{0})) = R(B_b + NV_b) b(j) + \sum R(C_i(B, A_i))

Cost(Connection(I_{0})) = \sum_i (d(P(A_i), P(B_l))(MIN(b(coding_{A_i, j})))

row.2 = (P(B_j)(MIN(Cost(Resources(I_{0}))) + Cost(Connection(I_{0}))))

end.

Appendix 9.3.4.7. Formal Description of the Algorithm 2: Optimizing all necessary resources parallel

IBridge2 (IBridge, Component, J_{1,in}, J_{1,out}, ..., J_{N,in}, J_{N,out}, Coding_{J_{1,in}}, Coding_{J_{N,out}},

Algo_{J_1}, ..., Algo_{J_N}, Network)

begin

*Initialize alternatives

do for all configurations I_{0,1}  \text{ \text{*I_{0,1} = (j, p, i, p_i, i_2, p_2, ..., i_N, p_N) representing all coding structures and locations}}

I_{0,1}.1 = j

FindOptimalLocation(I_{0,1})

do

*algorithm

optimum.1, optimum.2 = (j,il(MIN(Cost(I_{0,0})))

TryToLocateBridge(I_{optimum.1, optimum.2})

*alternative options
if Failure(I(optimum.1, optimum.2)) then
    j = optimum.1
    i = optimum.2
    row = I(j,i)
    case Failure of
        Bridge
            row.2 = N(row.2)
            Connection
            if ((row.error).eq.1).and.((n(row.(error+1))).eq.1)) then
                I(j,i+1) = (j, N(pj,i), 11a, p1a, i2a, p2a, ..., iNα, pNα),
            fi
            if ((row.error).eq.1) then
                I(j,i) = row\(row.error = row.error + 1)
                I(j,imax+1) := row\(row.error = 1, row.(error+1) = N(row.error))
            else
                I(j,i) = row\(row.error = row.error + 1)
            fi
        end
        goto *algorithm
    fi
do for all i
    i_Connection(Ai, Bj, coding_Ai, j, Network)
od
end.

FindoptimalLocation(I(j,i))

begin
    Cost(Resources(I(j,i))) = R(Bi + NVb) b(j) + Σi R(C1(B, Ai))
    Cost(Connection(I(j,i))) = Σi (d(P(Ai),P(Bj)))(MIN(b(coding_Ai, j)))
    row.2 = (P(Bj))k(MIN(Cost(Resources(I(j,i))) + Cost(Connection(I(j,i))))))
end.

Appendix 9.3.5. Other Options for Implicit Bridging (Component Mapping)

If it is desirable to include other configurations in the algorithm for the implicit bridging (component), the following routine can be followed:

1. First locate the bridge, using the connected parties as explicit endpoints.
2. Recursively locate the N-distributor and N-combiner for each party not connected to the bridge, using all other parties or the connections already made (for the bridge or previous other N-distributers and N-combiners) as explicit endpoints.
3. Repeat this for all possible coding structures of the bridges and combinations of parties (connected to the bridge versus not connected to the bridge).
4. Take the optimal configuration (yielding minimal cost) and try to implement it.
5. Upon failure, introduce an alternative according to the algorithm described algorithms for connections, distributers, combiners and bridges.
It is important to note that the location of the bridge is optimized first in this algorithm. This decision seems purely arbitrary. However, it is very likely that the bridge (with a relatively large number of inputs) is less available than the distributers and combiners, for which many alternatives are available as well. The algorithm evaluates all possible alternatives (if bridges with 0 to $N$ inputs and outputs are allowed). It is not guaranteed however, that the optimal solution is found.
Appendix 10. The MAGIC Global Modeling Overview

In this Appendix the MAGIC Global Modeling Overview, as given in [MAG06], page 27, is given.