FLEXPLANET, a Flexible Multi-Layer Network Design Tool

Zoltán Zsóka*, László Jereb†, Tamás Izsó*, Ferenc Unghváry†
zsoka@hit.bme.hu
*Budapest University of Technology and Economics, Department of Telecommunications
†University of West Hungary, Institute of Informatics and Economics

***Abstract*—The provision of services with complex multi-layer communication networks requires appropriate and generic network models describing accurately the existing networks and flexible softwares supporting the adaptation of the used algorithms to the new planning and analysis tasks. The paper introduces an object-oriented modeling and planning tool that is suitable to describe the existing multi-layer networks in detail, provides framework for modeling, dimensioning and analyzing new technological solutions, and for introducing new planning and evaluation approaches and algorithms. The framework is used at Hungarian Telekom, and its capabilities are validated with some illustrative examples.**

I. Motivation and preliminaries

The evolution of information services requires the continuous enhancement and development of the telecommunication infrastructure. This development is not a simple qualitative change, but in the last two decades several new communication technologies become available and many others were investigated as possible options. Both the available and the possible future technologies offer increasing capacities, however, instead of replacing each other step by step the different technologies compose complex multi-layer architecture and provide higher level and new networking and end-user services.

This always changing and evolving environment requires complex networking models that support not only the management of the different networking layers, but the administration, design and analysis of these multi-layer networks taking into consideration the resilience schemes provided by the different layers. Therefore, the provision of high-value services with complex multi-layer communication networks requires not only appropriate and generic network planning and analysis models that accurately describe the existing network structure and components, but the design and analysis software environment should be flexible to support the adaptation of the used algorithms to the new dimensioning and evaluation challenges.

The academic research group provides planning and analysis models and tools for the Hungarian Telekom for more than 20 years ([1], [2]). In the past 10 years this background describes year by year more precisely the current network status by taking information from the management systems. The role of this operational database is presented in details in [3].

The research work has been partially granted by OTKA Project 048985 and by the Foundation for the Development of Higher Education of Technology (15.12.2007). The work was supported in part by the Inter-University Cooperative Research Center for Telecommunications of Budapest University of Technology and Economics.

This paper focusses on the flexible network model and tool, and it is organized as follows:

- in Section II the relating works are briefly summarized,
- in Section III the structure of the design and analysis tool is introduced,
- Section IV presents the concept of the flexible network model and some details concerning its main components,
- Section V illustrates the application of the model, and finally
- in Section VI the paper is concluded and the planned further activities are highlighted.

II. Network modeling approaches

The network designing models can be classified according to several aspects. In the next subsections two of the interesting viewpoints will be emphasized: the generic models for different technologies provided by the telecommunication industry itself and the necessity of object-oriented programming in the network designing and analysis tools.

A. Multi-layer network models

Modern telecommunication networks are multi-layer networks since they are decomposable into a number of layer networks. The general concept of partitioning and layering are specified in recommendation [4]. Recommendations [5], [6], and [7] extend the adaptation of this concept to SDH, ATM, and optical networks, respectively.

A layer network topology is described as a graph that consists of nodes and edges. These nodes and edges represent the components of the networks, for example in [8] and [9]. In the abstract model each node has one or more roles e.g traffic generator, junction which correspond to network devices [10], [11], [12].

The partitioning aims to reflect the internal structure of a layer network [4]. This approach allows to hide the complexity of a network, and to handle a group of components as only one abstract network element called subnetwork.

B. Models and frameworks

According to one of the useful classifications these models and tools can be identified as (i) proprietary network models and (ii) frameworks (see [13]).

Proprietary network models are used by a vast number of network design tools to solve specific tasks or to describe specific network types.

The advantages of these models are

- the network model can be rather simple,
- relatively fast algorithms can be developed.

On the other hand, these models usually are not generic, and therefore, there are several disadvantages of these solutions as well. Some of them are as follows:

- even the well known algorithms (like Dijkstra) should be newly implemented in each tool,
- all algorithms used in the tool should also be newly implemented if the model is changed,
- in each case the developer has to know the detailed network model to develop a new algorithm.

The other group of the network designer tools is called **frameworks**. They use generic network models that can describe any current network types. In the early frameworks the network model and the algorithms are tightly coupled. Although the framework can handle most of the actual network types since it uses generic, multi-layer network model (e.g. [1]), it is difficult to develop a network specific algorithm because the algorithms work directly on the network model, and therefore, all developers have to know all of its details.

The second group of the frameworks (e.g. [14]) separates the network model and the algorithms. The network model describes a wide range of networks, while the algorithms have their own data formats to work on. The framework transforms the necessary part of the network model to the input data format of the algorithms and updates the network model by using the output of the algorithms.

The advantages of this architecture are as follows:

- the designers of the network model have to focus only on the network model, they do not need to consider the possible algorithms,
- the network models are relatively simple,
- the designers of the algorithms have to focus only on the algorithm,
- the input data formats of the algorithms can be optimized for the algorithms,
- existing algorithms can be used, since transformations are used between the model of network and the model (input and output data formats) of the algorithms.

The frameworks belonging to this group can apply the component-based development paradigm ([15]) that allows to separate network models and algorithms and to use existing algorithms without newly implementing them.

The third group of the frameworks (e.g. [16]) logically separates the network model and the algorithms. These network models usually are object oriented and provide some generic interface (like graph) for algorithms by inheritance and polymorphism. Although the algorithms are logically separated from the network model, they work physically on it via the interfaces developed by the network model designers. The advantages of this architecture are:

- the designers of the algorithms have to focus only the algorithm,
- there is no need for transformations between the network model and the algorithms,
- when the network model is changed but the interfaces are not, there is no need for new implementation of the algorithms.

The disadvantages of this architecture are:

- the designers of the network model have to consider the possible types of algorithms to create and implement interfaces for them,
- the network models are complex, since on the one hand they have to describe most of the current networks, and on the other hand, they have to implement the generic interfaces for the algorithms,
- the implementation of generic interfaces may be not optimal for some algorithms, thus using some of the algorithms is inefficient,
- when the generic interfaces are changed, the algorithms using these interfaces have to be newly implemented as well.

Modern framework software tools are object-oriented or component-based. Using object-oriented approach and design patterns [17] the network model can be flexible (easy to extend) and can provide generic interfaces for algorithms ([13]) in order to separate algorithms from the internal structures of the network model. The subnetwork approach of [4] is used in [8], [16] that describe the object-oriented network model as a part of a network design framework.

In the past 20 years we have followed the above-mentioned stages of software technology, and have experienced many of the described advantages and disadvantages. This paper provides a tool that belongs to the second group of frameworks, namely it is object-oriented, supports all current and possible future networks and supports short and long term network design and analysis.

III. Structure of the tool

A. Main objectives

The main aims in the development of the network design and analysis tool can be summarized as follows:

- The tool has to support the development and evaluation of an existing, concrete network. It has to be flexible enough to follow any changes in real telecommunication networking, therefore, any details of the existing network components that can be relevant in the provisioning, dimensioning or analysis processes should be captured by the model.
- On the other hand, by means of the tool future networking options are also investigated. Thus, there are needs for the abstraction in the model in order to avoid the description of the unnecessary or yet unknown details when strategic solutions are studied, or academic research is performed.
- The tool is a continuously evolving system, therefore in any stages of the software development, the already implemented functions shall be available immediately.

- Since the tool should be used for planning the development of the existing network or to investigate possible applications of future technological solutions where the limitations due to the existing network may have significant impact on the results, the software should work in both operator and academic environment and therefore, it should be portable to different platforms.
- Since many times to find efficient and/or already implemented dimensioning and analysis algorithms is the key issue, only public domain solutions and not commercial ones can be applied.
- The development, testing and running of the programs shall be performable on a normal PC without the direct use of external resources, e.g. operator's databases.

B. Main components of the software

The FLEXPLANET system consists of the following building blocks that are implemented as independent, but cooperating programs:

- database connectors,
- administrative frame for netmodel management,
- network design and analysis frame,
- graphical viewer of network netmodels.

Note that in our notation a *netmodel* is used as the model of a concrete network.

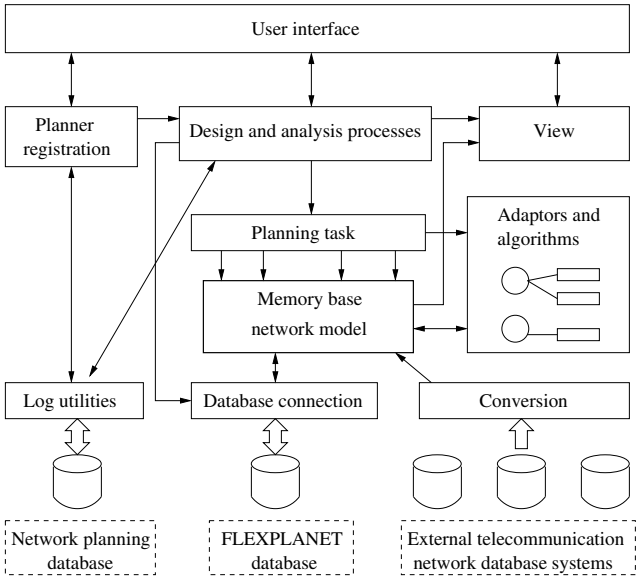

Fig. 1. Main components of the system

Figure 1 presents the functional components of the network design tool. The short description of the most important components can be given as:

- **Memory base network model** is a weighted point of the system (see Sect. IV for more details).
- **Database connection** maintains the synchrony between the inner FLEXPLANET netmodel(s) loaded in the memory and the physical storage of the netmodel's data.
- **Conversion** realizes the connection between the inner FLEXPLANET netmodel and the data extracted from external databases that describe operating networks.
- **Design and analysis processes** manage the activity of the planners on a netmodel in order to perform certain dimensioning and evaluation tasks.
- **Planning tasks** are the implemented methods that a planner can apply to modify the netmodel.
- **Adaptors and algorithms** realize the executable algorithms and transform the netmodel into data structure suited to the algorithms.
- **View** consists of several viewers and provides the presentation of information about the netmodels, planning or analysis results.
- **Planner registration** handles the identification, rights and activities of different planners. In the current version of the system only one planner is allowed.
- **Log** supports the tracing of planning processes and planner activities.

In a network design tool it is crucial how the data is obtained and maintained, that the planning processes working on. When the netmodel is built up on the basis of information coming from the database of an operating network, the system has to recognize and handle any inconsistencies occurring in that external database.

The FLEXPLANET system supports special reference netmodels that model the current state of the network database of the operator. This netmodel can be modified due to three causes, that may lead to collisions, since the modifications may not be synchronized:

- changes in the operating network,
- correction of missing or incorrect data,
- validation of the results of a planning activity.

Let us consider two modifying (e.g. planning) processes, that start with copies from the same stage of the reference netmodel and provide different ways of changing the data of a network component. A collision occurs when the reference model is modified according these parallel processes. Our system supports the copy-modify-merge model, i.e., the changes are calculated locally and modifications are performed in the reference netmodel as an operation, which is not dividable. Thus, in the case of parallel processes the one merging happens after the other and may cause contradictions. During the merging process an automatic or manual relaxation of the collisions can be done. The number of collisions is rather low, since the correction of one data entity has to be done only once and planning processes normally work on separated network zones. More details on this subject can be found in [3].

The topological and technological information about the netmodels can be presented also by the interactive graphical viewer. It applies a generic graphics library implemented on the very portable wxWidgets [18] basis. The main components of drawing are Draw, Layer, Shape and Visitor classes. The viewer is connected to the frame program that contains the inner FLEXPLANET netmodel as a socket IO technology

based client to a server. Thus the planner can follow very actively the impact of his work on the netmodel view.

C. Plannig task

A planning task may be built up from some elementary algorithm steps. Some of these algorithms require different data structures for the efficient execution. Since these structures are implemented independently from the network model structure the planning steps can be improved very effectively.

Planning starts with the definition of the set of the considered network elements and their adoption to the special data structures of the algorithms. The mutual mapping of the elements is indispensable. The algorithms are classified in FLEXPLANET and well known frameworks are chosen for their implementation:

- **Graph problems** are solved using the LEMON (Library of Efficient Models and Optimization in Networks) C++ template library developed by the Operations Research Department of the ELTE University [19],
- **LP optimization** is performed using LP_SOLVE [20], GLPK [21], CPLEX [22] or ILOG Concert Technology [23] when available,
- **Combinatoric problems** are solved using soft computing heuristics or the GECODE constraint programming library [24].

The results calculated by the algorithms will be set in the netmodel based on the mapping references.

IV. Flexible network model

A. Concept of the generic network model

The central element of the FLEXPLANET system structure is the flexible model that describes the network. This model is built up according to the following motivations and was implemented on object oriented basis:

- multi-layer network architectures have to be supported,
- flexible extension of the network description with new layers and technologies has to be provided,
- network information has to be stored in a design oriented manner,
- beside the network links, modeling node equipments and their internal configuration is required.

When an operating network is modeled we should not and can not store all the information that is available in the management database of a telecommunication service provider. The design (or analysis) oriented description of multi-layer networks rather focus on the hierarchical client-server connections between the resources of the different technological layers, the connections between the links and equipments, and the functional or structural connection of equipments in a network node. Thus, beside links and node equipments the FLEXPLANET contains also the following abstract components and properties of the modeled network:

- link rules that define the layers and their connectivity opportunities,
- node rules that identify the equipment, cards, ports etc. and their connectivity opportunities,
- node-link rules that describe the possible node-link connectivity opportunities.

Link rules are defined as disjoint sets of links with identical technological or logical functions, i.e., realizing a service-based layering according to the considered technologies and demands. Similarly, node rules are disjoint sets of equipments, that have the same type. The connectivity options are formal rules, that determine which client-server structures are allowed for the two abstract components. For instance, one can fix the number of links of layer A that can be multiplexed in a link of layer B or the number of ports of type P are integrated on a C type card. We may also define when the port P can be connected only to the links of a given layer L.

We emphasis again, that the design algorithms are implemented separately, using their own effective models. Matching the elements of the algorithm models and the descriptive netmodel is realized through suitable interfaces. Obviously, design algorithms consider technological constraints, but our aim is to pass this information in a generalized form.

B. Components of the model

The model is implemented according to the object oriented paradigm using the standard C++ language. This allows flexible improvement through the involved inheritance concept. In Figure 2 the UML diagram of the network model is presented. The following list defines the main classes:

- *NElement*: a basic type of network elements,
- *Link*: a component that realizes connection in a telecommunication network,
- *Node*: a component that realizes a logical or physical node in a telecommunication network,
- *NodeTempl*: equipment or site type,
- *N2N*: connectivity option for an equipment,
- *Level*: a logical or technological network layer,
- *Chn*: connectivity option of a network layer,
- *Net*: the modeled network,
- *Info*: additional information about the component.
- *Block*: set of network elements,
- *Pos*: identification of a channel inside a link.

Without loosing the generality and flexibility some specific classes from the *Link*, *Node* and *Block* classes are derived. Based on our experiences in network design, these special cases are identified as important ones, that should be handled rather differently, i.e., they implement different interfaces. Developers may add also new specialized classes if they are needed in further improvement of the tool.

Since the model itself shall be generic, the technology dependent information are stored as component properties (*Info*) instead of built in behaviour patterns for both the links and the nodes.

C. The concept of element hierarchy

On the one hand, the hierarchy of equipments and sites is realized by an array of children *Nodes* and an array of parent

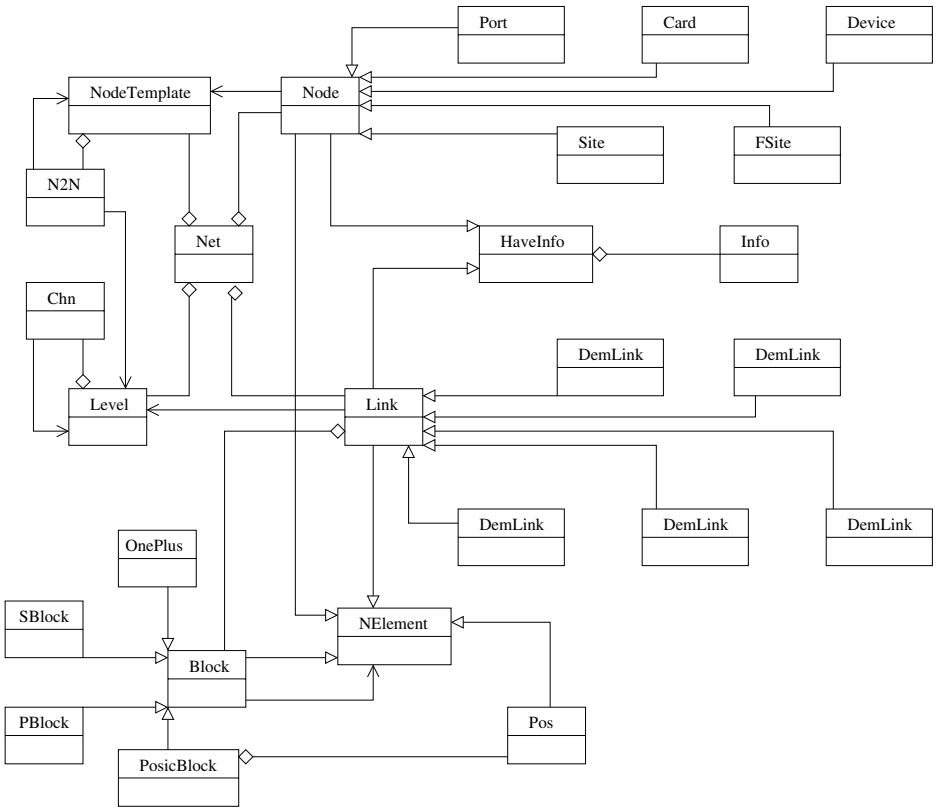

Fig. 2. UML diagramm of the descriptive network model

Nodes. These information are stored for each objects bringing some redundancy into the storage in order to speed up the traversal of such hierarchies.

On the other hand, the case of a network connection is implemented using *Blocks*. The classical approach for storing inter-layer information about links is to maintain an array of links. This array models a series of the serving (parents) or of the served (children) links, realizing the sections of the link or, respectively, the connections multiplexed on the link. Using this technique, the models for many networking issues are not straightforward, e.g. protection, restoration, multicast, etc.

Our approach instead, beside the *Links* considers both *Nodes* and further embedded *Blocks* in the hierarchy description of a *Link*. In order to distinct it from the classical hierarchy approach this very general concept is called the **Extension** of the *Link*. The extension is a *Block*, that contains an array of *NElements* and thus allows us to describe a wide range of complex hierarchical structures too. Another point for getting an entirely generalized description is, that in our model the point-to-point *Links* do not have a direct reference to their starting and ending node. Also the two end-nodes are embedded in the extension block.

Some species of blocks can play a general role in the description of a network, e.g. a serial or a parallel block. They demonstrate a set of elements, that are connected from the service of the served element point of view in a serial or parallel way respectively. On the other hand, special but often used types of block can be also identified as for instance those describing protected or load-balanced paths. If it has a reason and can be determined in the given technology, also the position of the serving link, that the client link uses, is described with blocks. This multiplexing information is embedded in a *PosicBlock*, that contains the serving link and the indices of its channels allocated for the client.

V. Demonstration of the network model

The paper [3] presents an illustrative case study of a complex planning and analysis task that is solved using the FLEXPLANET tool. The reliability analysis of a real size IP/MPLS core network is performed and the results are discussed.

In this section instead, we want to demonstrate the flexibility of the descriptive model through some application examples. The cases represent important tasks or networking issues. All of them can be caught using the FLEXPLANET model, but the real evaluation or planning work can be done only if the suitable planning steps are implemented, i.e., the model itself does not solves planning or analysis problems. The structures applied in the model have to be inserted by the tool-developers during the extension of the system in order to consider new technology or new networking concepts.

A. Path of a link

The classical and most important question in the link hierarchy description is how the path of a link is given. The extension concept differs from the very simple serial approach at some points.

As it was also mentioned in Sect. IV-C, the channel allocated on a server link for a client link is modeled as a special block in the extension of the client. This contains the channel positions and the server link.

The extension may contain nodes that are important from the link point of view. Starting (ending) node can be present in the block or it can be inherited from the first (last) server link. When a node in the extension does not accord to ending or starting node of the previous or next server link, it indicates a connection that is not included or not correctly described in the current netmodel. The same holds for the case of non-continuous serving link series, where some successive links are not neighboring ones.

These unidentified connections can be patch cables or switching connections between entry/exit points of the same or different technological layers inside an equipment. The details can be obtained by exploring the node hierarchy.

B. Dedicated path protection

Derived from the case of one path, it is easy to describe in the extension the multiple parallel paths of protection. We have between the starting and ending node a parallel block, that contains multiple blocks each one describing one of the paths.

Due to its wide application, we introduced also the special block type *OnePlus* for the 1+1 protection scheme. A small example can be seen in Figure 3.

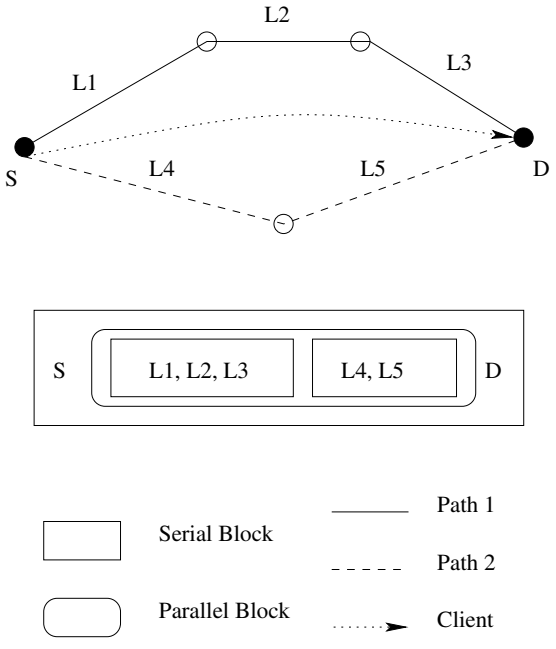

Fig. 3. Extension of a dedicated path protection

C. MPLS fast re-routing

The MPLS technology supports among other protection solutions the re-routing of packets to avoid the failed link. The link protection case can be modeled with a simple combination of serial and parallel blocks as it is illustrated in Figure 4.

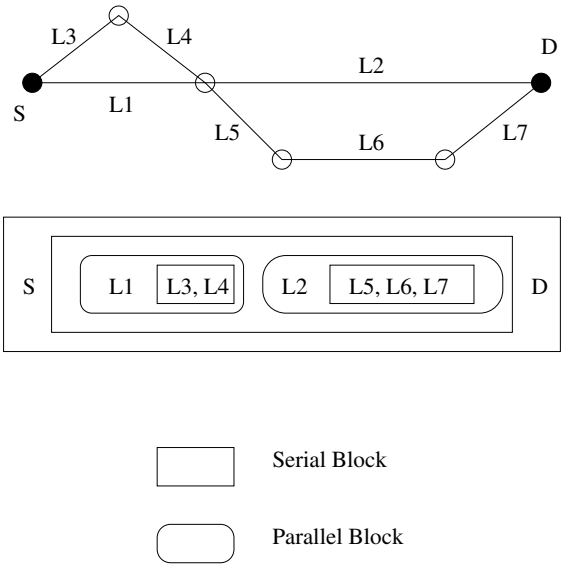

Fig. 4. Extension of a link protected tunnel

However, the node protection case and the use of general detour-paths needs more sophisticated blocks. The flexibility of the model makes it possible to insert such components.

D. Equipment reliability analysis

The reliability analysis of networks includes also derivation of parameters for equipments beside those calculated for the links. The factors in the computation are based on information that come from two not fully dependent characteristics.

On the one hand, the reliability parameters of a given equipment, card or port depends on its type, on its node template in our model. It may be either given, which means a general estimation or vendor specification, or be computed based on its complexity. The latter is in dependency with the defined connectivity options, stored in the model as *N2N* rules.

On the other hand, when the model for reliability calculation is built up, the current physical configuration of the network equipments has to be considered, in order to see the dependencies. This information is available through the node hierarchy of the netmodel's nodes.

E. Multicasting demands

A one-to-many connection demand is easily described using the link extension concept of FLEXPLANET. Since the extension is a quite general set of information, it contains also the end nodes of the traffic demand link. In the multicast case the extension block stores the source node and a parallel block. The latter contains all the destinations.

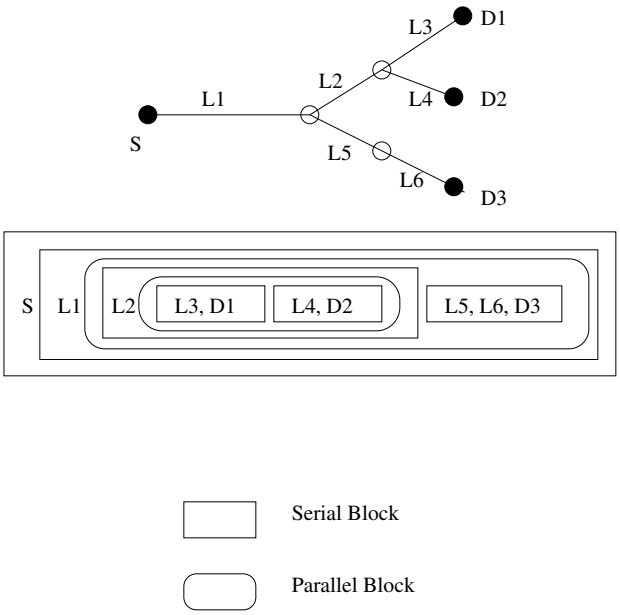

Fig. 5. Extension of a multicast demand

When such a demand gets routed there are multiple ways to model it, depending on the applied multicast protocol. The distribution tree can be described according to the recursive generation method, that starts from the source and inserts a new parallel block in the serial block of the current branch at each splitting nodes according to the split branches. Figure 5 shows a small example.

VI. Summary and conclusion

The rapid evolution of information services requires the continuous enhancement and development of the telecommunication infrastructure. Therefore, there is a permanent need for planning and analysis of existing networks, provisioning of new demands, installing new components in the network, and study possible networking options. The paper presented a modeling, design and analysis tool that makes it possible to consider real size operating networks using existing and future networking options in detail, and on the other hand, it is flexible enough to capture not only new networking technologies but new algorithmic solutions as well.

In the paper the structure of the software tool as well as the modeling concept are given. The software uses the object oriented paradigm, and some details on the components of the model are described. The capabilities of the tool are demonstrated by some examples including link path description, dedicated path protection, MPLS fast re-routing, equipment reliability analysis, and multicast demands.

The tool is introduced at Hungarian Telekom, where the previous versions of the software are applied for the everyday activities. The new software version is used for the modeling of the current network and its planning and analysis capabilities are permanently improved for handling new design and analysis tasks.

References

[1] L. Jereb, T. Jakab, M. Telek, A. Sipos, and G. Paksy, "Planet: A tool for telecommunication network planning and its application in Hungary," *Journal on Selected Areas in Communications*, vol. 12:7, pp. 1261–1272, 1994.

[2] L. Jereb, F. Unghváry, and T. Jakab, "A methodology for reliability analysis of multi-layer communication networks," *Optical Networks Magazine*, vol. 2, pp. 42–51, 2001.

[3] E. Babics, E. Csákány, T. Jakab, R. Konkoly, and L. Szandi, "Operational database-related multi-layer network modeling to support network development at Magyar Telekom," in *"13th International Telecommunications Network Strategy and Planning Symposium, NETWORKS2008 "*.

[4] "Generic functional architecture of transport networks," *ITU-T Recommendation G.805*, November 1995.

[5] "Architecture of transport networks based on the synchronous digital hierarchy (SDH)," *ITU-T Recommendation G.803*, March 1993.

[6] "Architecture of optical transport networks," *Draft ITU-T Recommendation G.872*, 1998.

[7] "Integrated services digital networks. overall network aspects and functions: functional architecture of transport networks based on ATM," *Draft ITU-T Recommendation I.326*, 1995.

[8] "OPNET modeler: Accelerating network R&D." OPNET Technologies Inc., 2004.

[9] S. Sengupta, A. Dupuy, J. Schwartz, O. Wolfson, and Y. Yemini, "The Netmate model for Network Management," in *IEEE Network Operations and Management Symposium (NOMS)*, (San Diego, CA, USA), February 1990.

[10] R. M. Redstall, J. R. Reid, R. Mardani, and J. Mellis, "OMNI – an object-based model of the access network infrastructure," *BT Technology Journal*, vol. 12, April 1996.

[11] I. Singh, "Cappuccino: An extensible planning tool for constraint-based ATM network design," Master's thesis.

[12] A. Varga and G. Pongor, "Flexible topology description language for simulation programs," in *The 9th European Simulation Symposium*, (Passau, Germany), pp. 225–229, October 1997.

[13] R. E. Johnson, "Frameworks = components + patterns – how frameworks compare to other obect-oriented reuse techniques," *Communications of the ACM*, vol. 40(10), pp. 39–42, October 1997.

[14] D. V. Tien, B. Kálmán, C. Király, and Z. Pándi, "A tool for the service planing and management of multi-layer networks," in *The 9th International Network Planning Symposium (Networks'2000)*, (Toronto, Canada), 2000.

[15] I. Crnkovic, "Component-based software engineering – new challenges in software development," in *The 25th International Conference on Information Technology Interfaces, 2003. ITI 2003*, (Cavtat, Croatia), June 2003.

[16] V. G. Fisher, "Design and implementation of FOONET – a framework for object-oriented network design," tech. rep., TU München, 1999.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley Publishing Company, 1995.

[18] J. Smart, K. Hock, and S. Csomor, "Cross-Platform GUI Programming with wxWidgets," October 2003. http://www.wxwidgets.org/.

[19] "LEMON documentation," 2007. Egerváry Combinatorial Optimization Research Group https://lemon.cs.elte.hu.

[20] M. Berkelaar, K. Eikland, and P. Notebaert, "Open source (Mixed-Integer) Linear Programming system," 2007. http://lpsolve.sourceforge.net.

[21] A. Makhorin, "GLPK (GNU Linear Programming Kit)," 2007. http://www.gnu.org/software/glpk/.

[22] ILOG, "ILOG CPLEX C++ API 9.0 Reference Manual," October 2003.

[23] ILOG, "ILOG Concert Technology 2.0 Reference Manual," October 2003.

[24] C. Schulte, G. Szokoli, G. Tack, M. Lagerkvist, N. Paltzer, and P. Pekczynski, "Generic constraint development enviroment (GECODE)," 2008. http://www.gecode.org.