Evolving system architectures for multimedia network design

E.A. Medova and J.E. Scott
Judge Institute of Management Studies
University of Cambridge
Trumpington Street, Cambridge, England CB2 1AG

July 14, 1998

Abstract

In our previous work we have developed a model to investigate capacity requirements and routing strategies for a network carrying hypothetical Asynchronous Transfer Mode (ATM) traffic. Our approach involves a hierarchy of design objectives associated with the respective network layers which constitutes a set of models – an Integrated Modelling System – with common data, 'solvers' and graphical user interface (GUI) operating under interactive control of the network planner. The system is used in the first instance for dimensioning link capacities which are optimally determined at the ATM layer by the Path and Capacity Allocation Model (PCA). Dimensioning at the optical transmission layer is based on a heuristic with solutions incorporated into the 2-level planning model solved by an LP or mixed integer programming solver.

The investigation of different network architectures, bounds on maximal link capacities, different forms of the objective function, and relations between cost and revenue coefficients are all accomplished using a modelling language (MODLER, AIMMS or XPRESS-MP) and an integration component which provides model management facilities. A set of user interface components have been implemented in the Java programming language, which allow visualization of solutions as they are calculated and interaction with the modelling process.

Our system may serve both as a network planning tool at the strategic level, and as a dynamic routing system at the operational level of management.

Introduction

Our work on an *Integrated Network Design System* (INDS) for multimedia high speed communication networks reflects the interests of British Telecom, who sponsored this work initially. Our present understanding of current requirements for the system architecture have grown through the company's position in global telecommunications. The principal tasks of the network design system developed are:

- capacity allocation for shared resources – different types of digital switches and communication links
- 'call' routing over an existing topology whose connectivity is enhanced through the concept of virtual connections.

Here the term *call* means a predefined mixture of multimedia traffic under the Asynchronous Transfer Mode (ATM) protocols. In our model three service classes of *constant bit rate* (CBR) sources – telephony, N-ISDN video and TV – and three service classes of *variable bit rate* (VBR) source – VBR video retrieval, Ethernet and high speed LAN – are multiplexed on *virtual paths* (VPs) or *virtual circuits* (VCs) at the so called *transport layer*. The underlying physical transmission network layer is based on the *Synchronous Digital Hierarchy* (SDH) protocols with a standard transport rate of 155Mb/s frames (STM-1) and their multiples (STM-N). Details about these transmission formats can be found in [17] and in previous publications related to this project [3, 4].

Here we would like to discuss our approach to the INDS system architecture which has been shaped by rapid technological developments and the competitiveness of the telecommunications industry. Debate is still going on between established telephone network operators offering all kinds of services and computer network (Internet) service providers claiming that they can handle any type of communication media. As a result, planning for future communications networks has become increasingly complex and weakly defined. In this project our problem formulations were often revised. Initially this was frustrating, but now this feature has finally been formalised as a principal property of our decision support system. Planning for multimedia networks involves problems with uncertainties of different natures: one is related to the natural stochasticity of traffic sources and the other – as noted – concerns modelling uncertainty reflecting technical implementations and management strategies translated into different network model formulations. Once modelling uncertainty is introduced, significant problems regarding capital investment decisions must be investigated. To handle both kinds of uncertainty we have developed a system consisting of a set of network models and a set of procedures for capacity allocation and routing with random demands. Our system may thus serve both as a network planning tool at the strategic level and as a dynamic routing system at the operational level of management.

The modelling is reviewed in Section 1 in order to demonstrate the need for easy reformulation of the initial problem using a *modelling language* [7]. Our approach involves a hierarchy of design objectives associated with respective network layers which constitutes a set of models with common data. In section 2 we present a current version of the Integrated Network Design System with common data, solvers and graphical user interface (GUI) evolving as a result of comparison and trials of different LP solvers, modelling languages and visualisation tools. The choices for modelling languages and solvers has been dictated by their availability in our environment and our established academic and business connections with their authors and does not indicate any specific preference in use. In the conclusion, we summarise our approaches and suggest the continuing development of an open system integrated through a flexible GUI.

1 Modelling

1.1 Review of PCA formulations

The *Path and Capacity Allocation* (PCA) [15] model serves as a principle component of INDS. Its formulation is generic and allows one to deal with a variety of practical requirements.

- Stochastic PCA

$$\min[\sum_{a\in A} b_a C_a - \sum_{w\in W} r_w \sum_{p\in P_w} \mathit{ess}\ \sup\{\mathbf{f}_p \mid \mathbf{f}_w \le D_w\}]$$

$$\text{s.t.}\quad \begin{array}{rlllll} P\{\sum_{p\in P_w}\mathbf{f}_p & \ge D_w\} & \le g_{burst} & & w & \in W \\ P\{\sum_{p\in Q_a}\mathbf{f}_p & \ge C_a\} & \le g_{burst} & & a & \in A \\ \mathbf{f}_p & \ge 0 & & p \in P_w & w & \in W \end{array}$$

- Deterministic PCA

$$\min[\sum_{a\in A} b_a C_a - \sum_{w\in W} r_w \sum_{p\in P_w} f_p]$$

$$\text{s.t.}\quad \begin{array}{rllll} \sum_{p\in P_w} f_p & = D_w & & w & \in W \\ \sum_{p\in Q_a} f_p & \le C_a & & a & \in A \\ C_a & \le C_A & & a & \in A \\ f_p & \ge 0 & p \in P_w & w & \in W \end{array}$$

In the stochastic PCA model *Grade of Service* (GoS) requirements are translated into *chance-constraints* in which D_w is a suitable *effective bandwidth* for multimedia calls between *origin-destination* (O-D) node pair $w \in W$, C_a is the *link-capacity* which guarantees the agreed GoS between user and network operator and C_A is the bound on *maximum link capacity.* Here b_a is the *cost of capacity* provisioning and r_w is the *unit revenue* of the traffic flows $f_p, p \in P_w, w \in W$.

In principle our deterministic PCA is a compact arc-path form *multicommodity flow problem* (MFP) in real variables [13]. The given reliability requirement for the network states that each pair $w \in W$ has a number of node/link disjoint paths $p \in P$ specified by the network operator. As the problem of finding these paths in a given network is computationally exponential, we precompute the required fixed number of paths to generate data matrices P_w and Q_a. The probabilistic calculations of the multilayer effective bandwidths are also performed separately by a specially designed algorithm described in [15]. As a significant amount of research is devoted to this topic with different views on the best implementation, the INDS system allows a choice for these calculations.

Although the reliability requirements introduce additional computational efforts, they simplify the resulting MFP by significantly reducing its size. National backbone telecommunications networks under consideration have less than 100 nodes with nearly sparse O-D demand specifications. This makes the solution time for practical problems very reasonable. The CPU times for the revenue maximization stage are shown in table 1, times are similar for minimization of the upper bound on flows.

PCA+switch revenue maximization with effective bandwidth calculation					
Network		Problem		Solution time (seconds)	
Name	BT	Rows	635	Effective b/w (caching)	5.23
Nodes	31	Columns	1376	Matrix generation	23.68
Links	70	non-zeros	9954	LP Solution time	0.75
OD-Pairs	216	density	1.14%		

Table 1: CPU times for IBM RS6000/590 running Dash Associates' XPRESS-MP under AIX 4.2. Matrix generation was performed by a *non-optimized* development version of mp-model.

Routing of the traffic over a network with unlimited capacity, i.e. over the shortest path between O-D nodes, can be considered as the first stage of network resource modelling. By examining the solution set C_a, the links of highest aggregated loads – *bottleneck* transmission links – can be identified. First stage routing solutions lead to highly uneven network link load distributions. As a policy, the network operator tries to 'spread' traffic across the network. We enforce this policy by a constraint on maximum link capacity C_A. A value for C_A is obtained using the *bottleneck model.*

At the next stage the value of maximum link capacity for given demand is fixed. The model will produce optimal traffic routings by splitting OD demand over all available paths with the most stringent constraints on maximum capacity installed and can be viewed as the final step in network optimization and traffic balancing – the *PCA+bottleneck model.*

Further analysis requires a more specific description of the network involving decisions about switching and buffering resources. The optimal node capacity can be chosen according to the optimal flow passing through the node, but current views on the switch size must be adjusted to future traffic expansion. At the SDH layer the network performs at much higher transmission rates. Translation of the VPI and VCI addresses is done according to the standards of the layer involved at the corresponding ATM or SDH switch. At the SDH layer the required ATM link capacities must be converted to SDH standards – i.e. STM-1, STM-4 or STM-16 – and these standards dictate integrality constraints on link capacity. In addition, the cost of switching equipment is highly nonlinear as it is decreasing in successive multiples for larger switch sizes and increasing levels of simplification. The corresponding PCA+ models are written in the system as *mixed integer programming* problems and are the topic of current research.

1.2 Hierarchy of design

The hierarchical design process is summarized below:

- Uncapacitated PCA

  Shortest path routing;

  Identification of heavily loaded links

- Bottleneck PCA

  Minimization of the bound on bottleneck links

- PCA + bottleneck

  Optimal routing over network with most stringent capacity constraints.

As we discussed earlier, the requirements for future networks are often redefined. For example, for each OD demand the overall traffic may be divided over two virtual paths, one for CBR sources – telephony, N-ISDN video and TV, and another for VBR sources – VBR video retrieval, Ethernet and high speed LAN. Each class may have a different procedure for the calculation of bandwidth requirements for different types of traffic.

In the case of the resource allocation for switches, the switched version of the PCA model (i.e. PCA+) must be used in a similar manner.

2 Software design

Our starting point for the INDS system was an *all UNIX* solution and is summarized in figure 1. We used both MODLER [10] and XPRESS-MP [2] modelling language for the modelling process, and found that there is little difference in the way that linear problems are formulated in these languages. MODLER, when used with ANALYZE [9] proved to be invaluable in the initial stages of the modelling process for the purposes of checking the correctness of the LP formulation. However, the commercial XPRESS outdid MODLER in speed, data handling flexibility and its ability to handle mixed integer problems which are the subject of our current research. There were several problems with the UNIX approach:

- The user was required to have a good working knowledge of UNIX, and an in-depth knowledge of the peculiarities of each modelling and solving environment employed, as well as a thorough knowledge of the problem formulation in order to use the system.

- As can be seen from the diagram, a dozen files can easily be involved in a single problem formulation. If small changes are made by the user (for instance, an adjustment of one OD-pair demand), either the number of files quickly explodes beyond management, or intermediate results are lost.

- There are no visualization facilities. It can be difficult to get an intuitive idea of what is going on from numerical tables of model results. For example, basic 'sanity checks', such as checking that there are no unconnected nodes, and checking that paths extend from their origin to their destination can be difficult and tedious.

In parallel, we also worked with Paragon Decision Technologies to produce a version that used the AIMMS environment [1], which consists of an integrated modelling language, solver and graphical user interface design system that runs under Microsoft Windows (see figure 2). AIMMS provides a user friendly environment in which both to do the modelling, and to design the user interface. However, this integration and ease of use came at a cost – we found it difficult to integrate AIMMS with custom software which would be necessary in the final system. For example, the effective bandwidth algorithm was placed in a Windows DLL with a proprietary interface – to which we had no access – so it was impossible to modify. The current (beta-test) AIMMS presentation facilities lack visualization tools that are specific to viewing network topologies, and we had no way of adding them. Later versions of AIMMS may address some of these limitations.

The Model Manager

At first, we partially solved the model management problem by automating the process shown in figure 1, and by endowing the management software with sufficient intelligence to notice changes between calculations, and recalculate only where necessary. To use the above example of a user changing one OD-pair demand, the model manager notices that it is unnecessary to recalculate shortest paths, and only one effective bandwidth need be calculated. If, as is likely, the demand has been changed to be the same as another demand for which the effective bandwidth has already been calculated, the data for that calculation is reused. [1] The model manager can also observe that the structure of the problem will not have changed, and that by identifying the constraint coefficients and RHS values that have changed (in this example there would be one RHS value), the change can be made to a copy of the matrix stored in memory from the previous run. This saves an expensive run through the model generation stage.

Figure 3 shows the model manager. The modelling process can be controlled by communicating with the manager by one of three interfaces. The command line interface is designed for non-interactive use, and allows the user to specify the problem and let it run. The *Network Dimensioning Protocol* (NDP) can be used to control the model manager by either direct interaction from the console by the user, or by communicating with another program (possibly on another machine) using the TCP/IP [2] interface. In this case the

[1]For the problem shown in table 1, although there are 216 OD-pairs, there are only 75 different values for bandwidth requirement, so this caching of expensive effective bandwidth calculations is also a saving while calculating the entire network.

[2] *TCP – Transmission control protocol.* This protocol is used to establish a two way connection oriented communications channel between two programs, possibly running on different machines. *IP – Internet protocol.* This is the protocol that defines how the TCP packets find their way between the machines.

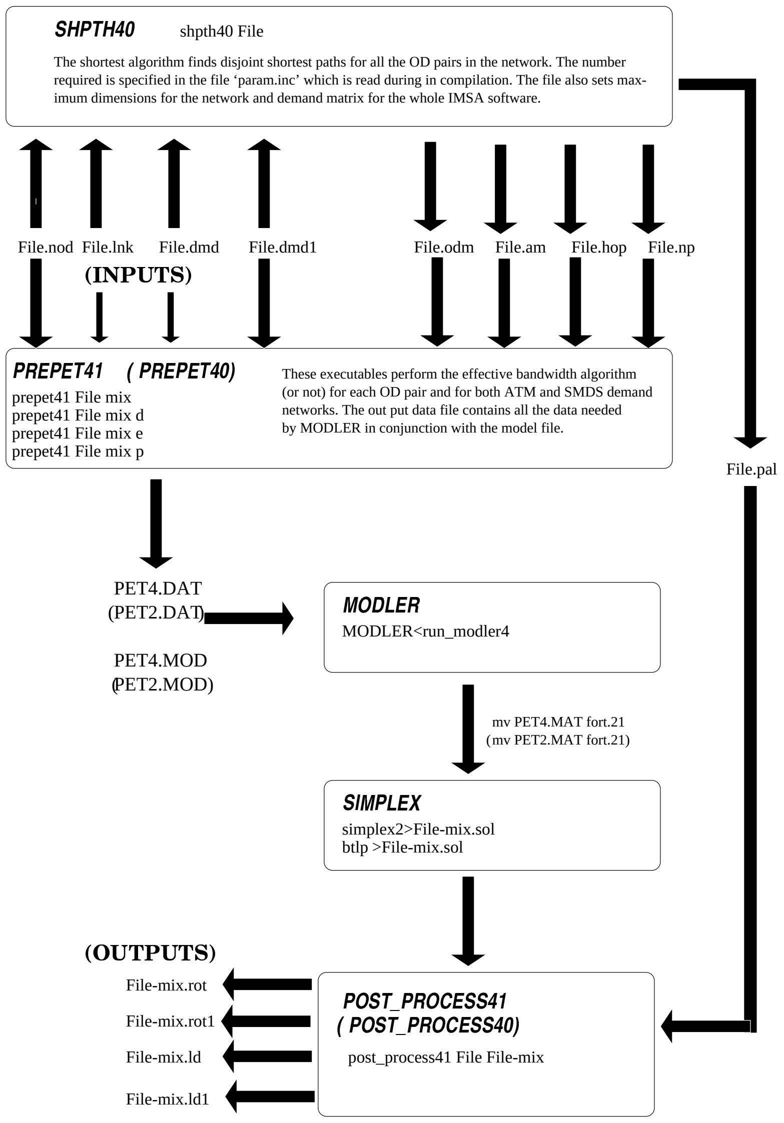
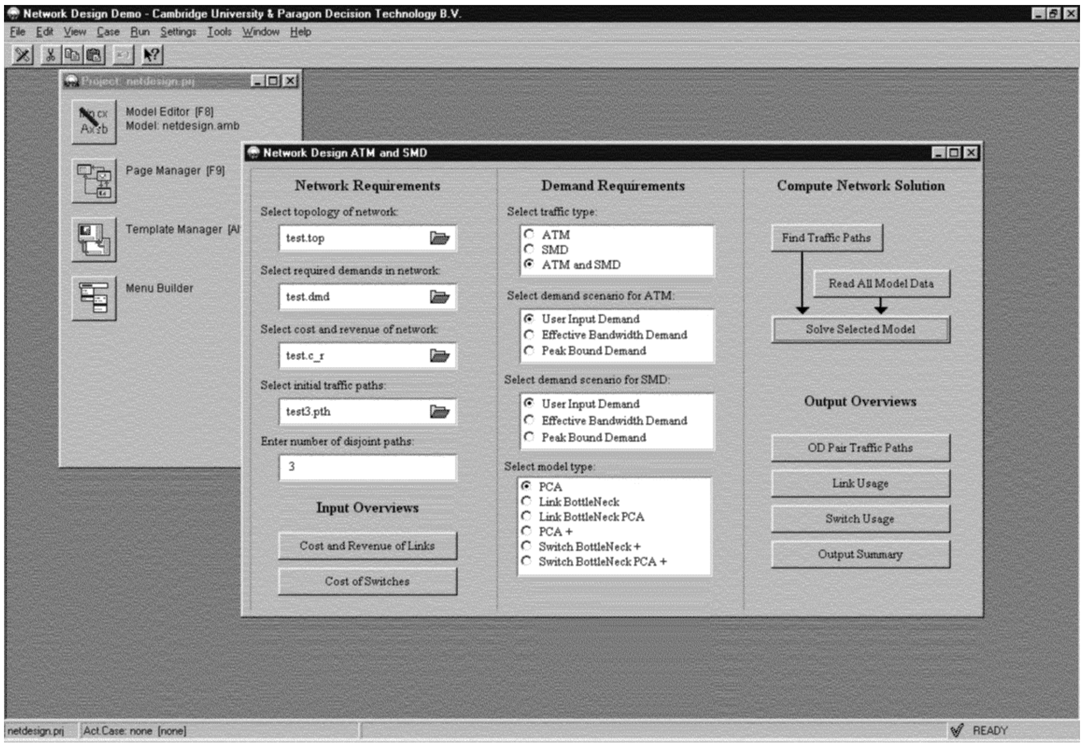

SHPTH40
shpth40 File
The shortest algorithm finds disjoint shortest paths for all the OD pairs in the network. The number required is specified in the file 'param.inc' which is read during in compilation. The file also sets maximum dimensions for the network and demand matrix for the whole IMSA software.
File.nod
File.lnk
File.dmd
File.dmd1
File.odm
File.am
File.hop
File.np
(INPUTS)
PREPET41 (PREPET40)
prepet41 File mix
prepet41 File mix d
prepet41 File mix e
prepet41 File mix p
These executables perform the effective bandwidth algorithm (or not) for each OD pair and for both ATM and SMDS demand networks. The out put data file contains all the data needed by MODLER in conjunction with the model file.
File.pal
PET4.DAT
(PET2.DAT)
PET4.MOD
(PET2.MOD)
MODLER
MODLER<run_modler4
mv PET4.MAT fort.21
(mv PET2.MAT fort.21)
SIMPLEX
simplex2>File-mix.sol
btlp >File-mix.sol
(OUTPUTS)
File-mix.rot
File-mix.rot1
File-mix.ld
File-mix.ld1
POST_PROCESS41
(POST_PROCESS40)
post_process41 File File-mix

Figure 1: Initial design of INDS. The network topology and demand information is submitted to `SHPTH40` where up to 3 shortest paths between each OD-pair are generated, and arc-path, node-path and OD-pair-path incidence matrices are generated for use in the PCA model. `PREPET41` calculates the effective bandwidth for each OD-pair, based on the contents of a 'mixfile' which describes the probabilistic behaviour of each traffic type. It then outputs a data file suitable for use by a modelling language (in this example `MODLER` is used). The modelling language creates a matrix which is the input to a simplex-based solver. The solution is then post-processed into a human readable format, making use of previously generated path data.



Network Design Demo - Cambridge University & Paragon Decision Technology B.V.

File Edit View Case Run Settings Tools Window Help

Project: netdesign.prj

Model Editor [F8]
Model: netdesign.amb

Page Manager [F9]

Template Manager [Al

Menu Builder

Network Design ATM and SMD

Network Requirements

Select topology of network: test.top

Select required demands in network: test.dmd

Select cost and revenue of network: test.c_r

Select initial traffic paths: test3.pth

Enter number of disjoint paths: 3

Input Overviews

Cost and Revenue of Links

Cost of Switches

Demand Requirements

Select traffic type: ATM SMD [✓ ATM and SMD]

Select demand scenario for ATM: [✓ User Input Demand] Effective Bandwidth Demand Peak Bound Demand

Select demand scenario for SMD: [✓ User Input Demand] Effective Bandwidth Demand Peak Bound Demand

Select model type: [✓ PCA] Link BottleNeck Link BottleNeck PCA PCA + Switch BottleNeck + Switch BottleNeck PCA +

Compute Network Solution

Find Traffic Paths

Read All Model Data

Solve Selected Model

Output Overviews

OD Pair Traffic Paths

Link Usage

Switch Usage

Output Summary

netdesign.prj Act.Case: none [none] READY

Figure 2: INDS AIMMS Control Panel. The user can specify the model using the selection boxes after supplying appropriate input files.

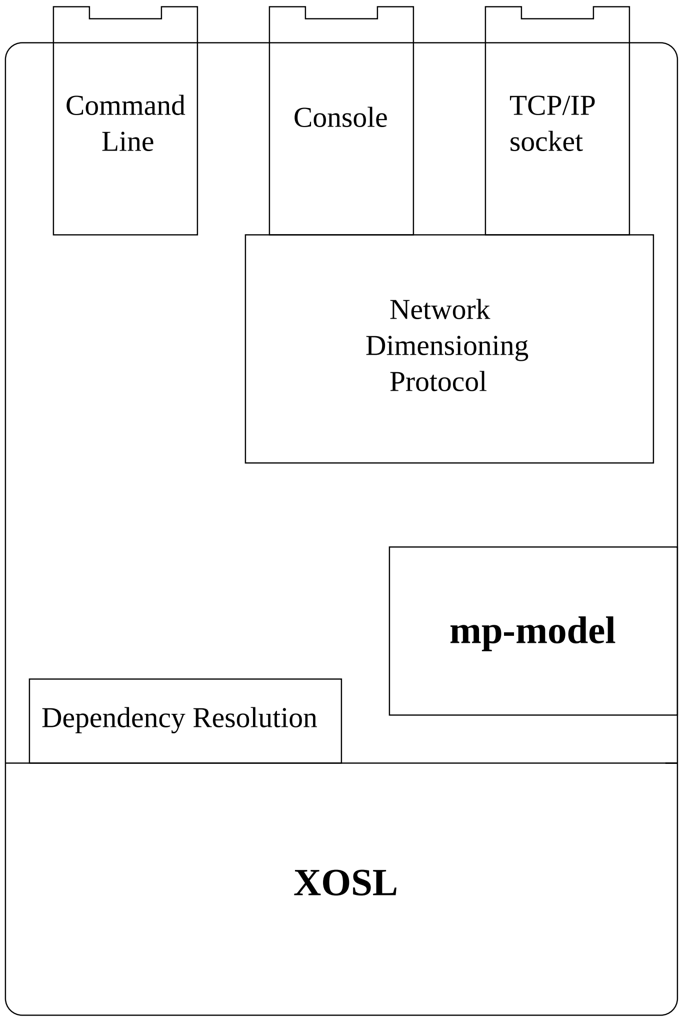


Figure 3: Network Dimensioning Model Manager.

model manager is effectively an internet *server* for network modelling, and the programs that connect to it are *clients*. Most often (and in our case) these clients would provide the user interface to the model. We considered separating the model manager from the user interface in this way to be necessary because of their contrasting requirements. Many modelling problems require state-of-the-art hardware to run, and often use legacy mathematical code. Mostly this means a UNIX/FORTRAN/C environment, whereas graphical user interfaces do not place much computational demand on their host, and are much easier to write in a high level language. Users have different preferences about what environment they work in, so the user interface should be easily portable between those environments. The ubiquity of the Internet and the TCP/IP protocol ensures that it is rarely difficult to connect the model manager and user interface together.

NDP is a simple command language that allows commands such as `SET DEMAND ATM 44 323.5` or `SHOW TOPO ATM` to control the actions of the manager and to read model data. One of the special features of NDP is that it allows the setting up of *observables*. These are model data which we wish to see every time they are changed, for example during the calculation of a solution. This makes the manager and its clients more complicated, as the manager has to keep track of observables while it is doing calculations, and the clients have to deal with unexpected observations while they are possibly doing other things, but this approach allows a high degree of interactivity between the model manager, the user interface and the user. The console interface is provided only to let the user debug the protocol by 'pretending' to be a client; so as to keep both clients and the server simple user friendliness is kept to a minimum.

The User Interface

Although the model generalizes well over different network management tasks, user interface requirements will differ considerably when targetted at different end users – consider the differences in interface requirements between an automatic dynamic router, a strategic planner who has to present to management, and the analyst/developer who requires complete flexibility. Because of this ambiguity of user requirements, we nominated ourselves (the analyst/developer) as our own targetted end user.

Our primary needs for a user interface were that the mechanics of model management became as transparent as possible when we were interested in the mathematical properties of the model, but that intermediate results of the modelling process were accessible. We had strong motivation to create a graphical user interface, as although current command-line based tools provide a rich and comprehensive modelling environment they have a steep learning curve. We were also interested in visualization techniques which would aid our understanding of the problem and its solutions.

Our requirements of portability and maintainability led us to choose Java [8] as our user interface implementation language. The user interface was implemented as a set of components connected by an *ObserverSocket/Observer* architecture. Any object in the system which changes can send a message to its ObserverSocket, and any Observers listening on that socket are made aware of the change. This enhances componentization of the user

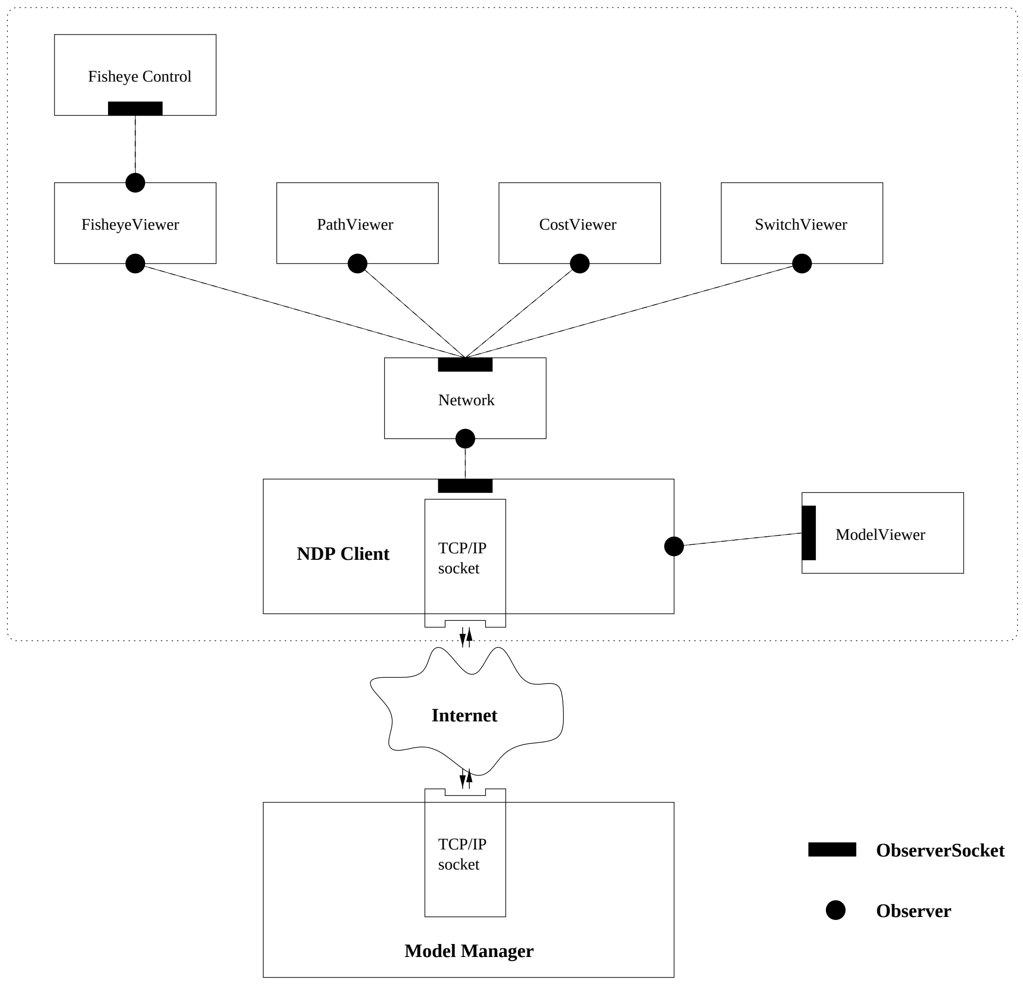
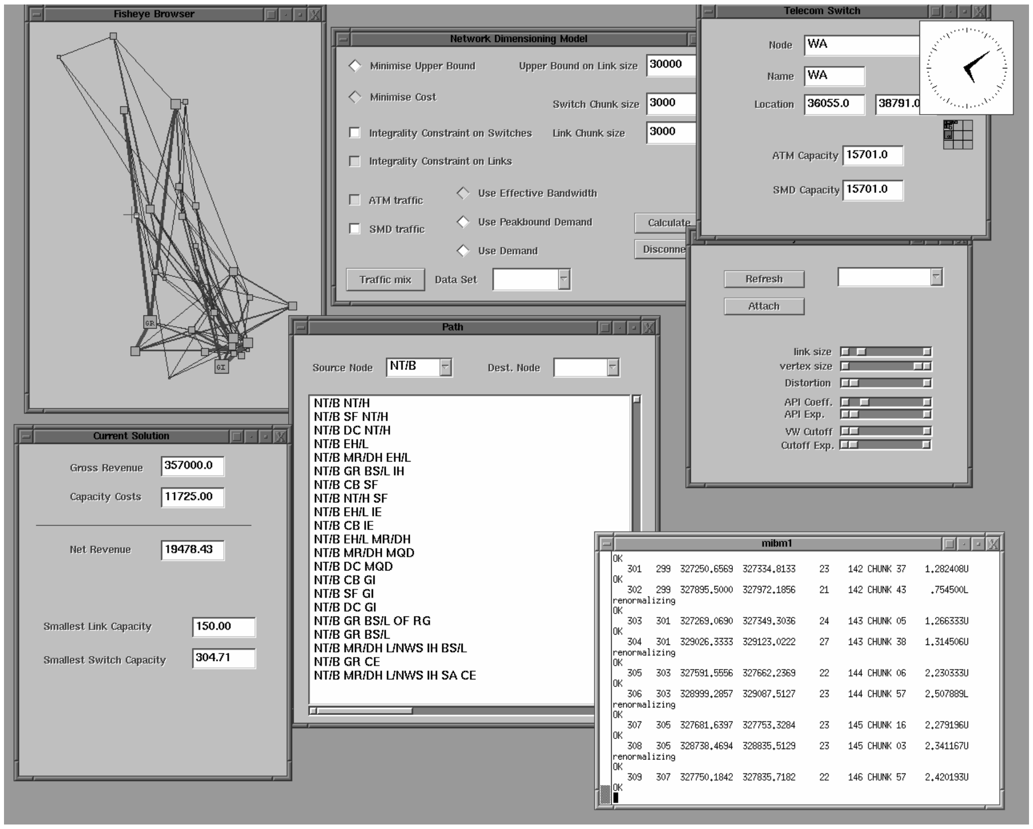


Figure 4: Complete System Architecture.

interface, as objects do not have to know about the things they affect. Figure 4 shows how the 'Network' object (which contains the network topology and dimensions) is kept up-to-date by observing the NDP Client, and the various specialized user interface objects observe the 'network' object to provide visualization. The NDP client observes the model viewer, which is where the user controls the model, and communicates appropriately with the model manager.

User Interface Components



Minimise Upper Bound
Minimise Cost
Integrality Constraint on Switches
Integrality Constraint on Links
ATM traffic
SMD traffic
Use Effective Bandwidth
Use Peakbound Demand
Use Demand
Upper Bound on Link size 30000
Switch Chunk size 3000
Link Chunk size 3000
Data Set

Node WA
Name WA
Location 36055.0 38791.0
ATM Capacity 15701.0
SMD Capacity 15701.0

link size
vertex size
Distortion
API Coeff.
API Exp.
VW Cutoff
Cutoff Exp.

Source Node NT/B
Dest. Node

NT/B NT/H
NT/B SF NT/H
NT/B DC NT/H
NT/B EH/L
NT/B MR/DH EH/L
NT/B GR BS/L IH
NT/B CB SF
NT/B NT/H SF
NT/B EH/L IE
NT/B CB IE
NT/B EH/L MR/DH
NT/B MR/DH MQD
NT/B DC MQD
NT/B CB GI
NT/B SF GI
NT/B DC GI
NT/B GR BS/L OF RG
NT/B GR BS/L
NT/B MR/DH L/NWS IH BS/L
NT/B GR CE
NT/B MR/DH L/NWS IH SA CE

Gross Revenue 357000.0
Capacity Costs 11725.00
Net Revenue 19478.43
Smallest Link Capacity 150.00
Smallest Switch Capacity 304.71

```
OK
   301   299   327250.6569   327334.8133     23     142 CHUNK 37    1.282408U
OK
   302   299   327895.5000   327972.1856     21     142 CHUNK 43     .754500L
renormalizing
OK
   303   301   327269.0690   327349.3036     24     143 CHUNK 05    1.266333U
OK
   304   301   329026.3333   329123.0222     27     143 CHUNK 38    1.314506U
renormalizing
OK
   305   303   327591.5556   327662.2369     22     144 CHUNK 06    2.230333U
OK
   306   303   328999.2857   329087.5127     23     144 CHUNK 57    2.507889L
renormalizing
OK
   307   305   327681.6397   327753.3284     23     145 CHUNK 16    2.279196U
OK
   308   305   328738.4694   328835.5129     23     145 CHUNK 03    2.341167U
renormalizing
OK
   309   307   327750.1842   327835.7182     22     146 CHUNK 57    2.420193U
OK
```

Figure 5: Going clockwise from the top left, this screenshot shows the 'fisheye' viewer, the model control window, the switch viewer showing details of the switch that is selected in the fisheye view, the fisheye control window, a dump of the communication between client and server, the path viewer showing all paths out of the Nottingham switch (NT/B) and a window showing the current costs and revenues.

Figure 5 shows the user interface components running in a UNIX environment. Because the code is written in Java, the client runs without change on Unix and Windows platforms. It is a matter of preference whether the user interface components adopt the 'look and feel' of the window system they are running on, or maintain a system independent style. We adopted the latter option, and used the BISS-AWT toolkit [16] in the construction of the interface as it offered a more mature development environment than the standard Java user interface toolkit alone at the time. Later versions of Java and commercial 'visual' development environments now make user interface programming in Java as straightforward and as powerful as in environments such as Visual Basic or Delphi.

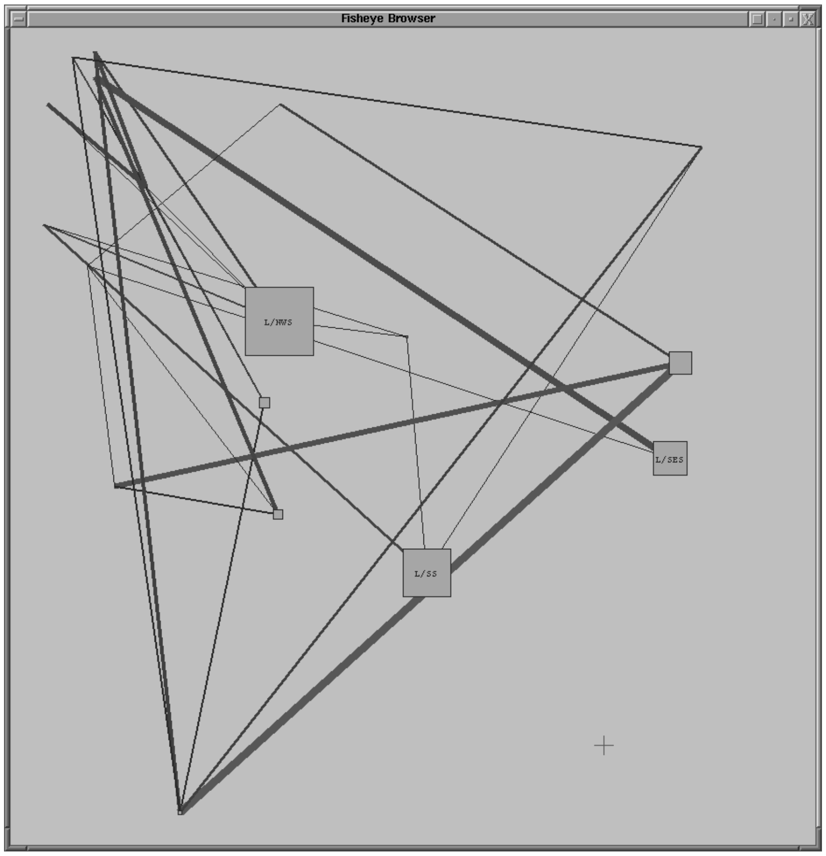


Figure 6: The fisheye viewer showing the user looking at the London area with only major connections shown, while in figure 5 the entire network is shown. This component observes the 'network' component, so as the dimensions of the network change during the course of solution the display is updated.

The fisheye component is an implementation of the ideas presented in [18]. It uses a topology preserving transform which simulates the effect of a 'fisheye' camera lens, in which objects near the focal point are scaled up and objects further away reduced, so that the user can see local detail of the network while keeping global context. All objects on the view are given a 'visual worthiness' value between 0 and 1 based on their distance from the focus and an *a priori* importance such as capacity. Choosing a cutoff point using the fisheye control, objects of lower importance can be removed from the view to reduce clutter (see figure 6). See [14] for a good survey of network visualization techniques.

In our current research on integrality constraints on links and switch capacity, as intermediate solutions are produced (for example during branch-and-bound search) they update the capacities on the fisheye view, so an intuitive idea of how the search space is being explored can be gained. By using concurrent 'threads' of execution, the program enables the user to manipulate the view while it is being updated by the solver.

3 Conclusion

The focus of this paper has been the issues surrounding implementation of the Integrated Network Design System in which stochastic calculations are combined with modern mathematical programming tools. The advantages of using a modelling language have been established for a long time [5, 11] in a variety of applications. In spite of this fact such tools are rarely applied in the telecommunications industry.

Our current system demonstrates the importance of introducing optimization modelling tools to network design, interfaced using an effective and portable GUI. The complex issue of integrating different network layers may be accomplished through a hierarchy of design procedures and modifications of the basic underlying PCA model. Our current choice of XPRESS-MP was motivated by our interest in solving hard mixed integer programming formulations of the SDH dimensioning problem.

Future Goals

By keeping each of our user interface components self-contained, it is our intention that future versions of the software will comform to one of the standard 'component' architecures such as JavaBeans, Microsoft ActiveX or OpenDoc [12]. This will give us a toolkit of components which can be easily integrated with third party products such as spreadsheets or databases. We will also look at the possibility of using a relational database schema [6] to formalize the storage of data connected with the model, as although our model manager minimizes the amount of user interaction to manage one version of the model, multiple variations of the model data still have to be managed manually.

Acknowledgements

The work reported here was partially supported by BT Laboratories, Martlesham Heath, Ipswich, England and the UK EPSRC. We are grateful to Professor Harvey Greenberg for his long term support and advice regarding his mathematical modelling and analysis software. We would also like to thank Professor Jan Bisschop for his collaboration and continuing interest in our work. Thanks are also due to our colleagues in the Finance and Systems Research Group at the Judge Institute.

References

[1] Bisschop, J. and Entriken, R. (1993). *Aimms – The Modeling System.* Paragon Decision Technology B.V., Haarlem, NL.

[2] Dash Associates Limited, Blisworth, Northants, UK (1998). *XPRESS-MP User Guide.* 10th edition.

[3] Dempster, M., Medova, E., Azmoodeh, H., Key, P. and Sargood, S. (1996). Design and Control of ATM/SDH Networks. in *Proceedings of the 4th International Conference on Telecommunication Systems*, Vanderbilt University, Nashville, 259–270.

[4] Dempster, M., Medova, E. and Thompson, R. (1997). A Stochastic Programming Approach to Network Planning. Volume 2b of *Teletraffic Contributions for the Information Age.* Proceedings 15th ITC, Washington DC, USA. Elsevier, Amsterdam, 329–341.

[5] Fourer, R. (1983). Modeling Languages versus Matrix Generators for Linear Programming. *ACM Transactions on Mathematical Software*, **9** 143–183.

[6] Fourer, R. (1997). Database structures for mathematical programming models. *Decision Support Systems*, **20** 317–344.

[7] Fourer, R., Gay, D. and Kernighan, B. (1990). A modeling language for mathematical programming. *Management Science*, **36** 519–554.

[8] Gosling, J. and McGilton, H. (1996). *The Java Language Environment.* White paper, Sun Microsystems.

[9] Greenberg, H. (1992). *A Primer for ANALYZE: A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions.* Mathematics Department, University of Colorado-Denver.

[10] Greenberg, H. (1993). *Modeling by Object-Driven Linear Elemental Relations: A User's Guide for MODLER.* Kluwer Academic Publishers.

[11] Greenberg, H. and Murphy, F. (1995). Views of mathematical proramming models and their instances. *Decision Support Systems*, **13** 3–34.

[12] Hamilton, G. (1997). *JavaBeans.* White paper, Sun Microsystems.

[13] Hu, J. (1970). *Integer Programming and Network Flows.* Addison-Wesley Publishing Company, Inc.

[14] Jones, C. (1997). *Visualization and Optimization.* Kluwer Academic Publishers.

[15] Medova, E. (1998). Chance-constrained stochastic programming for integrated services network management. *Annals of Operations Research* (forthcoming).

[16] Mehlitz, P. and Mehlitz, J. (1997). *The BISS AWT Framework.* Technical report, BISS GmbH.

[17] Nortel (1991). *Synchronous Transmission Systems.* Northern Telecom Europe Ltd, London.

[18] Sarkar, M. and Brown, M. (1992). Graphical Fisheye Views of Graphs. *Proceedings of the ACM SIGCHI'92 Conference on Human Factors in Computing Systems.*