[13] D. Bienstock, S. Chopra, O. Gunluk and C. Tsai, "Minimum cost capacity installation for multicommodity network flows," (1995) to appear, *Mathematical Programming* (Special issue on Computational Integer Programming).

[14] D. Bienstock and G. Muratore,"Experiments with a network design algorithm using e-approximate linear programs," (1996).

[15] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York (1988).

[16] T. Magnanti, P. Mirchandani and R. Vachani, "The convex hull of two core capacitated network design problems," *Mathematical Programming*, (1993) 233-250.

[17] T. Magnanti, P. Mirchandani and R. Vachani, "Modeling and solving the two facility capacitated network loading problem," INFORMS *Operations Research* 43, (1995), 142-157.

[18] T. Magnanti and Y. Wang,

[19] Y. Pochet and L.A. Wolsey, "Network design with divisible capacities: Aggregated flow and knapsack subproblems," (1992) *Proceedings IPCO2* 150-164, Pittsburgh.

[20] M. Stoer and G. Dahl, "A polyhedral approach to multicommodity survivable network design," Technical Report, University of Oslo, Department of Informatics (1995).

[21] V. Chvatal, A greedy heuristic for the set-covering problem, *Math. Oper. Research* (1975) 233-235.

[22] ILOG-Cplex Division, Incline Village, NV 89451.

- Costs sharply escalate as engineering is made more conservative, while under-engineering results in a significant increase in service risk without an appreciable reduction in capacity costs in most cases.
- Edge switching provides additional opportunities to realize costs savings through multiplexing and, in a high demand scenario, costs were reduced by as much as 40% relative to the baseline network.
- The impacts of strong communities of interest is to diminish the effectiveness of multiplexing ratios in reducing network capacity costs.
- Multiplexing is robust, in the sense that it makes a network insensitive to errors made in estimating demand levels and characteristics.

Future work will continue the effort begun here, with a goal of understanding the value and improving the stature, utilization and impact of network design in the evolving broadband networking.

REFERENCES

[1] I. Norros, "A storage model with self-similar input," *Queueing Systems*, 16 (1994) pp 387-396.

[2] W. Leland, M. Taqqu, W .Willinger, "On the self-similar nature of Ethernet traffic," *IEEE/ACM Transactions on Networking*, ? (1994) pp1-15.

[3] V. Benes, *General Stochastic Processes in the Theory of Queues*, Addison Wesley (1963).

[4] O. Narayan, "Exact asymptotic queue length distribution for fractional Brownian traffic," manuscript, 1997.

[5] A. Erramilli, W. Willinger, and O. Narayan, "Experimental queueing analysis with long- range dependent packet traffic," *IEEE Transactions on Networking*, April 1996.

[6] T. Carpenter, D. Heyman, I. Saniee, "Studies of random demands on network costs," to appear in *Telecommunication Systems*, manuscript, January 1996.

[7] I. Saniee, A. Erramilli, C. Pack, "The economics of statistical multiplexing in broad-band networks," Teletraffic Contributions for the Information Age, *Proceedings of ITC15*, Washington DC (1997), 815-824.

[8] D. Alevras, M. Grötschel, and R. Wessäly, "A Network Dimensioning Tool," Preprint SC 96-49, Konrad-Zuse-Zentrum für Informationstechnik Berlin (1996).

[9] A. Balakrishnan, T. Magnanti and R. Wong, "A dual-ascent procedure for large-scale uncapacitated network design," INFORMS *Operations Research* 37 (1989) 716-740.

[10] F. Barahona,"Network design using cut inequalities," *SIAM Journal on Optimization* 6 (1996) 823-837.

[11] D. Bienstock ,"Experiments with a network design algorithm using e-approximate linear programs," (1996).

[12] D. Bienstock and O. Gunluk, "Capacitated network design - polyhedral structure and computation," INFORMS *Journal on Computing* (1996) 243-260.

Test Case (5 runs for each)	CPU time in Seconds
3C0E.demL	10
3C14E.demL	20
N17C0E.demH	1200
N3C0E.demH	30
N3C14E.demH	50
N17C0E.demL	40
N3C0E.demL	10
N3C14E.demL	10

Table 7.1. Performance of the heuristic on 60 sample networks

0 edge,c. 3 core & 14 edge for each of two cases of 1. low demand and 2. high demand for each of two cases of 1. low community of interest and 2 high community of interest. For each case, 5 runs were conducted that varied the statistical multiplexing gains for all services from 0 to 20 in multiples of 5. A comparison of Table 7.1 with Table 6.1 reveals that the best results and the fastest runs are obtained when there are few switches and the total demand is high compared to the modular capacity of the transport systems where the solution was at least within 0.2%-1.7% of the optimum and run times where 10-30 seconds. The largest gap occurred when demands were low and there were 17 switch candidates, in addition to the 173 integer variables representing the transport selection on each link. The gaps appear not to depend on the degree of community of interest in the demand matrix. The CPU times are seen to be mostly confined to 1-40 seconds on a 100 MHz Sun Spark20 with 96 M of RAM but in the case of 17 backbone switches with the large demands with large multiplexing gains runs times reached 20 minutes (4 cases in the total of 60 experiments run). All the computations reported here used the commercial linear programming package Cplex 3.0 [22].

8. CONCLUSIONS

We draw two categories of conclusions from the computational experiments reported in the previous section: first those relating to the *quality* of the Integer programming approximations and heuristics used, and second regarding the engineering insight these solutions provided.

8.1 Conclusions relating to the integer programming heuristics

8.2 Conclusions relating to the engineering impact of network design alternatives

The following are the significant conclusions of the analyses:

- Statistical multiplexing can significantly reduce networking costs; in the case studies considered here, the cost savings ranged from 10%-40%.
- These savings in networking costs can be realized using modest levels of multiplexing gain at which costs and risks to performance can be balanced.

statistical multiplexing acts as a buffer that absorbs demand fluctuations. As in the previous case, this kind of robustness is obtained by overengineered.

These two notions of robustness were used in two sets of runs of the ATM design tool for the low demand scenario considered in the previous section. Figures 6.4 show the results. Data set 1 on the left shows the *modification* cost histogram when the network is originally designed for a modest level of multiplexing gain (5) for a representative value for each demand. As demands randomly change from half to twice their representative values, the modification costs as a percentage of total design cost is reported. Data set 2 corresponds to an identical arrangement with the difference that the network is originally designed for a much more aggressive level of multiplexing gain (15). Clearly, the network designed for data set 2 is lower in cost initially than that of data set 1 and this difference is under 10% of the total cost of network for data set 1 (as inferred from Figure6.4). However, as can be seen, the modification costs for network of data set 1 hardly ever exceed 5% of the original design, whereas the modification costs of network of data set 2 *average* around 22%.

We conclude that the network designed at a *modest* level of statistical multiplexing is more robust with respect to both criteria considered than the network designed at the more aggressive multiplexing gain and thus should result in lower *total costs* when uncertainties are significant.

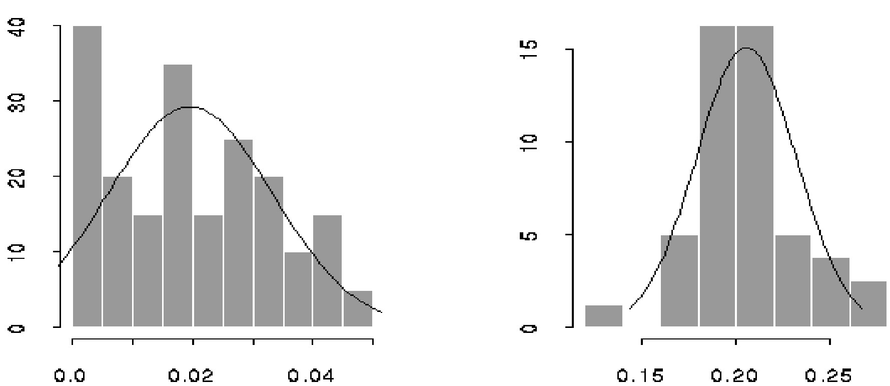

Figure 6-4. Modification costs as % of total cost for network over-designed (left) and network under-designed (right)

7. COMPUTATIONAL TIMES

The following table shows the computational times for the heuristics outlined in section 5 when applied to the 12 cases in representing choices of a. 17 core and 0 edge, b. 3 core and

Test Case (5 runs for each)	CPU time in Seconds
17C0E.demH	1100
3C0E.demH	10
3C14E.demH	40
17C0E.demL	40

6.2 Impacts of edge switching

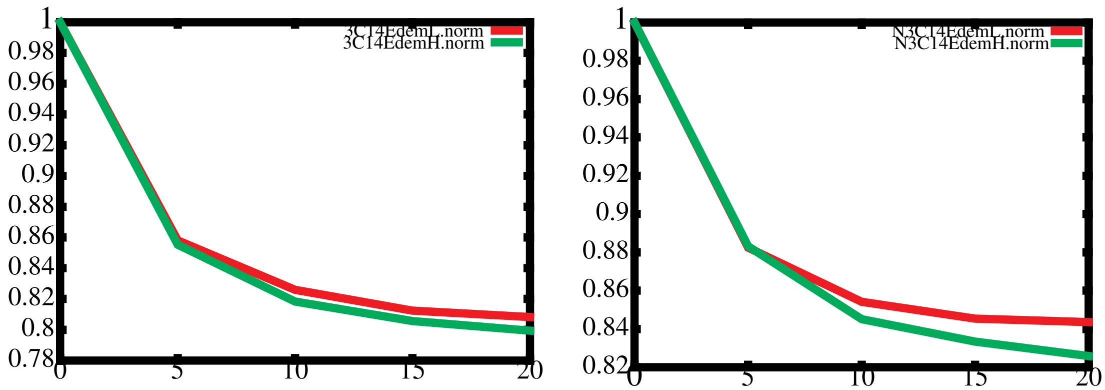

Figure 6-3. Capacity costs vs statmux gains for low (B) and high demand (A) scenarios, where costs are expressed as a % peak rate cases of corresponding scenarios in Figure 6-2.

Figure 6-3. shows plots of normalized network costs under low and high demand and high and low communities of interest. Network costs here consist of candidate backbone switches at 3 nodes and 14 candidate edge switches plus transport and plug-ins elsewhere in the network. The possible use of edge-switches provides an additional opportunity for realizing multiplexing gains, in the transport of traffic from the demand end points to the backbone switches. In general, feasible multiplexing gains will vary at different places in the network, e.g., they may be less for the trunks interconnecting edge switches and backbone switches than those interconnecting backbone switches. This is a conservative assumption if the minimum of the feasible set of multiplexing gains across all newtork resources is used.

The general trend of the plots is unchanged. As before, high communities of interest *reduce* the capacity cost benefits due to multiplexing. This is not surprising since less aggregation occurs when there is a community of interest. As a function of multiplexing gains, the savings in capacity costs are in the range of 10%-20% in the "good engineering" region for the low and high communities of interest, respectively. Again, as in the previous case, the cost benefits of multiplexing is greater for the case of higher demands.

6.3 Robustness

There are numerous interpretations of the desirable property of *robustness*. We focus on two useful interpretations. First, we look for robustness in the sense that a network design should be such that the demand forecast errors have a small impact on the total network modification costs. That is, the additional capacity cost to meet the unexpected demand is low for a robust network. Thus a network that requires only a 5% of its original cost to accommodate 40% change in demand forecast is robust compared to another network where the corresponding change is 30%. Clearly, this kind of robustness can be obtained by over design, which involves an initial design with low utilization to leave room for uncertainties. The second notion of robustness we use is with respect to statistical multiplexing: we call a network robust in this sense if a small percentage change in modification costs results when an increase in demands is unexpected *and* (then) statistical multiplexing is utilized via traffic management to minimize re-arrangement and modification costs. In this case

6.1 Cost-Benefits of multiplexing with backbone switching

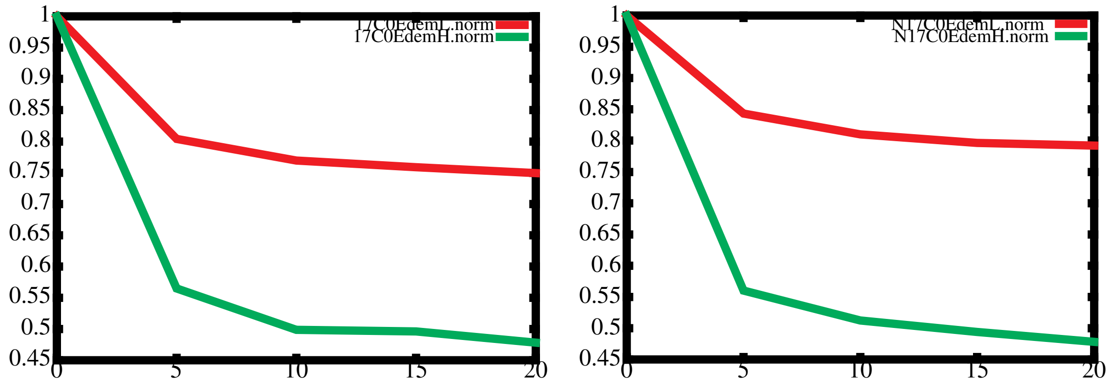

Figure 6-2. Networking capacity costs as a function of multiplexing gain for low (upper) and high (lower) demand scenarios and for high Community of Interest (right) and no CoI (left)

Figure 6-2 shows plots of normalized network costs under low and high demand and high and low communities of interest. Network costs here consist of candidate backbone switching at 17 nodes plus transport and plug-ins elsewhere in the network. Costs at different levels of multiplexing are expressed as a percentage relative to the cost of peak-rate engineering so that 100% corresponds to a multiplexing gain of 1. In other words, in what follows it is assumed that *the multiplexing gain of N means that the trunk between each pair of switches is dimensioned at 100/N% of the linear sum of the peak rates of circuits aggregated over that trunk.* Further, multiplexing gains are assumed the same for all services across the network for simplicity. As expected, capacity costs decrease as multiplexing gains. Overall, it is observed from Fig 6-2 that statistical multiplexing can cut capacity costs in the range of 20% for the low demand scenario and about 45% for the high demand scenario, these being close for high or low communities of interest. The plots can be broadly divided into three regions:

Over-engineering spanning multiplexing gains of 1-5, in this example, where the network is designed with caution, and good levels of service are obtained at the risk of high capacity costs. In particular, costs can escalate dramatically in the extreme case of peak-rate engineering as shown by the steep gradient of all curves near gain of 1 in Figure 6-2.

Under-engineering spanning multiplexing gains in excess of 10 for this example, where the network is designed with the intention of aggressive utilization, but may in fact may be characterized by "diminishing returns". Figure 6-2 shows under engineering does not necessarily achieve significant reductions in the capacity costs while potentially increasing the risks to service integrity. The *insensitivity* of the capacity costs to increasing multiplexing gains in this region is a critical fact for network design.

Good Engineering: Spanning the range of 5-10 multiplexing gains for this example, where one attempts to balance capacity cost and service quality. At the *low end of this region,* we see that even modest levels of statistical multiplexing can lead to significant decreases in the capacity costs. In this region, even without adequate traffic measurements to estimate the Hurst and peakedness parameters, one is able to sharply reduce the capacity costs without jeopardizing quality of service.

Name	Mux Gain	Cost	Lower Bound	Gap (%)
	5	182658148	1.776982e+08	2.79
	10	174758308	1.698382e+08	2.89
	15	172083788	1.671315e+08	2.96
	20	170724108	1.657781e+08	2.98
17C0EdemL	1	25835720	2.251866e+07	14.73
	5	20776220	1.785854e+07	16.33
	10	19880340	1.692941e+07	17.43
	15	19608880	1.663494e+07	17.87
	20	19363340	1.645184e+07	17.69
3C14EdemL	1	24908320	2.276211e+07	9.42
	5	20776220	1.889386e+07	8.88
	10	20234780	1.864251e+07	8.54
	15	20133020	1.848785e+07	8.89
	20	21374040	1.967986e+07	8.60
N17C0EdemH	1	205919108	1.963680e+08	4.86
	5	115404828	1.104120e+08	4.52
	10	105519388	9.908060e+07	6.49
	15	101786208	9.526875e+07	6.84
	20	98467448	9.336282e+07	5.46
N3C14EdemL	1	25397220	2.349870e+07	8.07
	5	22413320	2.073808e+07	8.07
	10	21694351	1.991536e+07	8.93
	15	21424011	1.963244e+07	9.12
	20	21471995	1.951604e+07	10.02
N17C0EdemL	1	26222540	2.315757e+07	13.23
	5	22106080	1.911556e+07	15.64
	10	21238460	1.820813e+07	16.64
	15	20878180	1.793695e+07	16.39
	20	20774260	1.775300e+07	17.01
N3C14EdemL	1	36516660	3.574813e+07	2.149
	5	28969620	2.824012e+07	2.58
	10	28689580	2.675412e+07	7.23
	15	28116060	2.736929e+07	2.72
	20	27814780	2.713635e+07	2.50

Table 6.1 Computational Results

low levels of admissible multiplexing encourage edge switch placements while large amounts of long-haul inter-nodal circuits might encourage direct trunking to the backbone switches unless a large level of statistical multiplexing is permissible. Finally, for each of the above four cases, two *architectural alternatives* are considered: (a) all 17 core nodes are candidates switches and (b) 3 out of the 17 core nodes with highest demands are candidate *backbone switches* and the remaining 14 nodes are candidate *access switches*. We distinguish these cases by the naming convention described in the following table.

6. Computations and interpretation of results

We will consider the total cost of deploying a high-speed data communications network as allowable multiplexing gains are varied. In effect, we are using multiplexing gains as a surrogate for evaluating the capacity costs associated with the QoS provided in the network. via direct use of (3) since 1. at present time we do not have accurate measurements of the traffic parameters of the said services and 2. we wish to *quantify* the economic thresholds of multiplexing gain directly. As multiplexing gains are varied, the full spectrum of trade-off choices -- from peak-rate over-engineering to mean-rate under-engineering -- are covered. Thus a network designed with overall multiplexing gain of 4 is one in which the inter-switch trunks are a quarter of the size they would be if constant bit-rate peak-rate allocations were used. As the levels of multiplexing are changed for each architecture (see Table 5.1), a potentially different solution is obtained. Figures 6-1 illustrates a solution consisting of 4 fully-connected backbone switches at which demands from all nodes terminate.

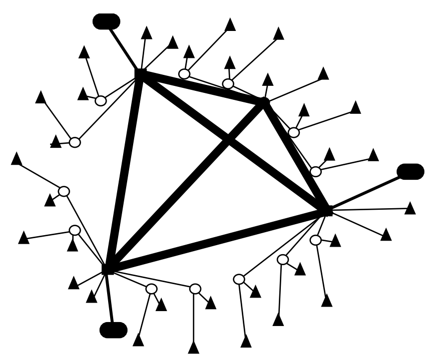

Figure 6-1. A 4-Backbone-Switch Design (Cf. Fig 5-1)

Table 6.1 shows the results of the 5 runs for each *architecture* where the level of statistical multiplexing was varied from 1 to 1/20 of the peak rate in increments of 5. Columns 2, 3 and 4 show the total cost of each network, together with lower bounds obtained from the solution heuristics outlined in Section[?], and the percentage gap between the solution and the lower bound as measured by 100*(solution - lower bound/lower bound).

Name	Mux Gain	Cost	Lower Bound	Gap (%)
17C0EdemH	1	204543588	1.961922e+08	4.25
	5	115543268	1.056880e+08	9.32
	10	102035088	9.392014e+07	8.64
	16	101506228	8.999295e+07	12.79
	20	97803788	8.802936e+07	11.10
3C14EdemH	1	213574248	2.078708e+08	2.74

Also, for each scenario two sub-scenarios are considered: Case 1, or *low community of interest*, where demands are almost uniformly distributed to all other nodes within the each of the three node categories subject to the constraints described above and Case 2, or *high community of interest*, where the same total level of demand is now highly concentrated between certain set of nodes.

Name	Only Backbone/Core & Access Switches (B/E)	Large/Small Demands (L/S)	Community of Interest/ Distributed Demands (C/D)
17C0EdemH	B	L	D
3C14EdemH	E	L	D
N17C0EdemH	B	L	C
N3C14EdemH	E	L	C
17C0EdemL	B	S	D
3C14EdemL	E	S	D
N17C0EdemL	B	S	C
N3C14EdemL	E	S	C

Table 5.1. Naming Conventions for the Case Studies

5.2 An overview of the assumptions

It is assumed that all circuits will be routed through at least 1 and at most 4 switches of which at most 2 may be backbone switches and at most 2 may be edge switches. Thus the simplest route uses just one edge or backbone switch while a more complex route uses 1 edge switch followed by 2 backbone switches and 1 other edge switch before the circuit reaches its destination. The backbone switch has total port capacity equivalent to 20000 mb/s for any combination of DS1 and DS3 ports while the edge switch has port capacity of 3 DS3s, 84 DS1s or 2016 DS0 ports. The switches are assumed to cost $1,450,000 and $90,000 for backbone and edge switching, respectively. It is assumed that access termination costs on the edge switches have negligible costs while those for the backbone backbone switch run at $100 for DS1 and $5,000 for DS3 or equivalent terminations. Transport costs are present worth of monthly tariffs over a 3-year period and are assumed to be $1,500 for DS1, $25000 for DS3. The cost of media (or the mileage dependent cost of transport) averaged and discounted for 3 years is $124/mile.

5.3 An overview of the architecture alternatives

The optimization code takes node-pair point-to-point service-based loads as input and attempts to route them on *permanent virtual circuits* in such a way as to minimize the total of switching, transport and media costs subject to meeting demands at the required grade-of-service. The result is a set of access links connecting end nodes to edge/core switches and trunks connecting switches to each other. The number and location of switches, the topological configuration of the desired network and the dimensioning of the access and trunks all depend on the specifics such as the admissible levels of statistical multiplexing, the QoS as well as transport modularities and switch port capacities. The optimal overall design for a snapshot of the demand forecasts clearly depends also on the characteristics of demand data as well. For example, a large amount of localized or intra-nodal demands with

One can prove that, under appropriate assumptions, the worst-case behavior of this heuristic is bounded as in [21].

5. SAMPLE DATA AND COMPUTATIONS

Since performance of optimization algorithms is best measured in the context of *real* data, as opposed to *randomly generated* examples [?], we focus on realistic variations on a real network. Thus instead of generating 60 random graphs with random demands to test our code, we study 60 variations with real *engineering* significance on a real network. (The engineering significance of these runs are reported in the next section.) The network used for the computational experiments reported in Section? is abstracted from a real data network consisting of 54 nodes and 173 physical links. Of the 54 nodes a total of 17 are fully connected and are existing or candidate locations for backbone or edge switch placements and 3 are hand-off locations for circuits destined for nodes outside the network. The links vary in length from a few dozen feet to about 40 miles. The services provisioned in this network are four variables bit rate data services with peak rates 64 kb/s and 1.5 mb/s for Frame Relay, 1.5 mb/s and 45 mb/s for switched multi-megabit data and video services. The mean utilization of each circuit for each service ranges from 5% to 27% of the peak rate. The *Hurst* parameter of the services ranges in [0.6,0.8] with the *peakedness factor* in [10,10000]. Each node belongs to one of three node categories of low, medium, and high demand, where half the total number of circuits from each node are routed to nodes outside the network via the three hand-off nodes. Of the circuits with both ends in the network 6% are intra-nodal while 94% have distinct origins and destinations within the network.

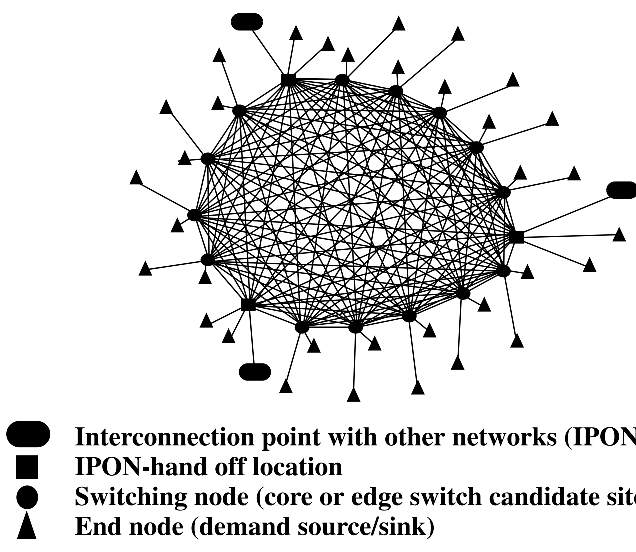

Figure 5-1. A logical view of the network studies

5.1 An Overview of Demand Scenarios

The non-zero point-to-point demands for the number of circuits for each service between each pair of nodes varies from a few 45 mb/s SMDS or video to a few dozen 64 kb/s FR and are forecast over a period of 5 years for the *low demand scenario*. For the *high demand scenario* the same demands are 10 times as large (see discussion in Section ??).

$$Minimize \quad \sum_u c_u y_u + \sum_{l \in L} t_u w_u + \sum_e k_e f_e \tag{22}$$

Subject to: (23)

$$\sum_e f_e + \sum_u w_u \geq d \tag{24}$$

$$y_u \geq t_u + \sum_e f_e \qquad \forall u \tag{25}$$

$$f \geq 0, \quad w \geq 0, \quad y \in Z \tag{26}$$

The intuition behind this formulation is as follows. The network is an abstraction of our logical network where we only keep the switch location candidate nodes and the links between them. Our (one) commodity can be routed using logical paths that use one switch or two switches; we may view this as using one node or one edge of the graph G. Constraint (24) states that our routing satisfies the demand (variable f_e is the amount routed using edge *e;* in terms of the real problem this corresponds to using one switch at each of the endnodes of e. Similarly, variable w_u corresponds to a logical path that only uses one switch located at u). Constraint (25) states that we place enough capacity at each node -- here we are assuming that each switch has unit capacity, which can be obtained by rescaling the data (i.e. the quantity d may be fractional -- also when we have many demands of many services the true form of (25) is a bit more complex). The objective contains a (true) switching cost, plus routing costs (which are a linearized version of all costs we ignore). Finally, we have ommitted the survivability constraints for simplicity.

When the network layout is large, this simplified problem may still be too large for exact solution or even to attempt a cutting-plane type approach. However, a careful look at this problem reveals that it has features reminiscent of a set-covering problem. In our approach we have adapted the classical "greedy" heuristic to our particular problem (see [21]).

Our implementation of the greedy heuristic essentially discretizes the variables w_u and f_e. At each iteration, the heuristic chooses one of these variables, corresponding to a demand that is either not yet fullfilled or to a survivability constraint not yet satisfied. The variable will be chosen to minimize a penalty value, roughly the ratio of cost to infeasibility reduction among all such variables. To compute this ratio we also choose the amount by which we will increase this variable from its current value (this is where the discretization comes in). This may also result in an increase of a variable y_u.

Somewhat simplifying our rules, given one such variable (either a w_u or a f_e) there are two cases:

(a) all inequalities (25) involving this variable are slack, the increment is the minimum value needed to make any such inequality tight, or the least amount needed to satisfy the constraint (24) (if still unsatisfied) or the least amount need to satisfy any (violated) survivability constraint. In this case the penalty for z is simply its unit cost.

(b) some inequality (25), say for node u, is currently tight. In this case we increase y_u by one unit, we increase z in a way similar to that used in case (a) and the penalty is the ratio of the total cost involved in this increase (switch plus flow) to an estimate on the total potential decrease of infeasibility achieved by this increment.

one variable to round **up** (based on a criterion such as least increase in cost per unit of additional capacity). If we examine the variables in the order just described we are always guaranteed to have a feasible integral solution. Occasionally, if the chosen variable actually has very small value, we first attempt to round it down, and compare the round down and round up cases as discussed above.

After this stage of rounding only the link capacity variables remain. At this point, however, we have already decided where and how many switches and/or concentrators to place, and the feasible routings are very limited. Our last rounding stage examines the links of the network (essentially in order of load, or ratio of transmission cost to load) and solves a knapsack problem to decide the optimal capacity allocation.

4.3 The preprocessing stage

As has been alluded to above, the formulation can be very complex and also very large. For a large enough network with enough demands and enough core switch and concentrator candidate nodes, one might obtain a formulation with several million variables and hundreds of thousands of constraints, and which is prohibitively difficult to solve as a linear program. On the other hand, it is likely that in the final solution only a relatively small number of paths will be used.

This suggests that a column generation (used to generate paths; i.e. **not** used to generate integer feasible solutions) approach might be a good idea. However, based on the complexity of the formulation, we decided that this approach would also be prohibitively expensive. Note that and one necessary ingredient for a successful column generation approach is a set of good heuristics, and we felt we did not have such heuristics (the “obvious” ones did not work -- more on this below). Finally, we also felt that a sophisticated column generation approach would compromise the robustness that we seek throughout.

Instead, we chose the following approach. First, build a family of “obvious” paths -- these are essentially shortest paths and cost-effective paths (that take into account some linearized version of the transmission and switching costs incurred by the path) and also “diverse” paths. To this family of paths we add those produced by a more sophisticated procedure described next.

The spirit of the procedure is to simplify the problem so as to model the salient feature (switching) using integer variables and linearizing all other costs. Roughly stated, in the simplified problem we route commodities using paths that go through one or two switches, the switches have normal capacity and cost, and all other costs are linearized. An exact or approximate solution to the simplified problem can then be used to restrict attention to a subset of core switch locations (i.e. we ignore those switch locations not used in the solution to the simplified problem) and to those paths that are consistent with the paths found in the solution.

In our description of the simplified problem, we will assume that we have exactly one demand to be routed in the ATM problem, say with d units of demand. The simplified problem is as follows: we are given an undirected graph G. We have to choose, for each edge e a continuous nonnegative variable f_e, for each node u a continuous nonnegative variable w_u, and for each node u an integral nonnegative variable y_u, so as to solve

One final word about cuts is that some care has to be exercised as the cuts can lead to some numerical instability . This is particularly the case with very large problems (where the cuts are dense and lead to large memory requirements) and problems where the data is not very well scaled (in which case the LP-solver may experience instability).

4.2 The rounding and branching stage

After adding a suitable number of cuts, our algorithm moves into a branching and variable-fixing stage. This stage itself is subdivided into several parts.

In the first part the algorithm tries to round to integers (primarily, round down) some of the critical variables, in particular the variables concerning core switches. We use the following generic approach. To fix ideas, let A denote a set of nodes that can contain core switches, and denote

$$S = \sum_{a \in A} y_a \tag{20}$$

Suppose that in the current LP-relaxation S takes a very small value. Then we would guess that in a good design none of the nodes of A will actually contain switches. In other word, we would like to round down (and fix to zero) these y_a variables. Unfortunately, it may be the case that (probably due to poor scaling of the data) this is not a good idea. To test this rounding systematically, we essentially branch on the variable S. One node (the "down" node) has $S = 0$, and the other node (the "up" node) has the inequality

$$\sum_{a \in A} y_a \geq 1 \tag{21}$$

added to the formulation. Our approach immediately chooses one of the two nodes and fathoms (removes) the other. The algorithm will favor the down node in all cases except when either (a) the down node is infeasible, or (b) both nodes are feasible and the cost (value of LP-relaxation) at the down node is at least a certain threshold percentage more expensive than the cost at the up node. Typically we have used values such as 10% or even 20% as the threshold.

In early testing of our algorithm, we found this procedure useful enough that we apply it for many subsets A, not just those with the value S small. As with inequality (19) we try this rounding step on all singletons and complements of singletons, and the right-hand side of (21) is of course $\lceil S \rceil$.

In many cases, after applying this procedure many times we obtain a formulation that is significantly simplified and also stronger. One observation is that many datasets start out with a large number of nodes that can be candidates for hosting a core switch -- but in the final solution only a small subset is actually used. Our procedure tries to guess such a set, and at the very least it strengthens the formulation with several new inequalities of the form (21). One effect of the rounding down of core switch variables is that it tends to reduce the complexity of the routing -- fewer routing choices are available, which limits the possible paths that can be used, and this in turn results in increased values for the other integer variables (especially the link transmission system variables).

Our next stage of rounding is completely standard. Here we look at the integer variables in categories: first the plug-in variables, then any remaining (non-rounded) core switch variables, then the concentrator or edge-switch variables. Each iteration of rounding chooses

of the cutset. For example, if the load on a certain edge is *500* units and the facility with least cost per unit capacity can carry *10000* units of flow, then that facility (on that edge) will be given weight 0.05 on that edge. Inequalities of the form (18) (with B the throughput of the most efficient facility type) will cut off this kind of fractional point. In general, given a cutset C, we employ the above procedure for all the facility types in the cutset.

Recall that we obtained (17) by using appropriate values to multiply the inequalities (15) and then add them to the sum of the inequalities (14). By choosing these multipliers appropriately, so that in (18) the only flows with positive coefficients are at value *0* in the current LP-relaxation, one essentially obtains an inequality that lower bounds the total capacity of C.

Another strategy that we used seeks to extend the way that (13) strengthens (12). For example, suppose we have a cutset C made up of nodes where core switches can be installed. We want to add an inequality that places a lower bound on the number of switches that we install in nodes of C.

Suppose that we solve a linear program with all the constraints in the model, but whose objective function is the number of switches installed in C, and that the value of this LP is 2.4. Then we know that in any integer-feasible solution the number of switches in C is at least 3, and we can add this inequality to the formulation. In general, for a given cutset C, we get a valid inequality of the form

$$\sum_{a \in C} y_a \geq \lceil F(C) \rceil \qquad (19)$$

where *F(C)* is the value of the linear program whose objective function is the left-hand side of (19). This simple procedure strengthens the formulation in the same form that (13) strengthens (12). Notice that by using the LP-relaxation we possibly obtain a stronger right-hand side than by a direct analysis of the problem, except possibly a fairly complex analysis. Morevoer, this procedure is inherently robust: should the problem formulation change somewhat, a complex analysis as was just mentioned may no longer yield a strong, or even valid, bound, but the LP-based procedure remains valid.

A critical step in generating inequalities (18) or (19) is the choice of the cutset C. Following the work of other researchers, and, in particular, our work in [12] and [13], we used an approach that combines some enumeration and also the choice of cutsets that are "tight" in the current LP-relaxation. Another trick is to maintain a list or "cache" of cutsets that were used in previous inequalities to test them against the current LP-relaxation. Yet another trick consists in "growing" a cutset that was previously used by adding a node (and the reverse procedure, as well).

In addition, for inequalities of type (19) we can explicitly enumerate certain *sets* (not just cutsets). For example, we single out one node, or the set of all nodes minus one node, or a pair of nodes, or the set of all nodes minus one pair of nodes. In each case the solution of an appropriate linear program yields an inequality of type (19). The reason we use sets and not just cutsets is that this procedure takes into account the survivability features of the model, when present (for a more in-depth look at survivability issues, see [14] or [18]).

This procedure can be surprisingly effective. Part of the reason for this is that in many cases the data is such that the number of core switches to be installed overall is not very large, and yet the cost of these switches will be a principal component of the total. Inequalities (19) can dramatically improve the lower estimate on the minimum cost provided by the LP-relaxation, as well as aid our heuristics (see below).

Let C be a cutset (in the logical network), potentially made up of both nodes and links. From our formulation we can **abstract** a system of constraints of the type:

$$\sum_{i \in M(e)} m_i z_i + A(e) \geq \sum_{k \in K} \lambda_e^k f_e^k \qquad \forall e \in C \tag{14}$$

$$\sum_{e \in C} f_e^k \geq d(k) \qquad \forall k \in K \tag{15}$$

$$f_e^k \geq 0 \quad \forall e \in C, \forall k \in K, \quad z_i \in Z, \forall e \in C, \forall i \in M(e) \tag{16}$$

Here K is a set of commodities separated by C, i.e. they must cross C. For k e K, $d(k)$ is the total demand of commodity k, and if e e C, f_e^k is the flow of commodity k in element e of C. Thus constraint (15) states that we route a sufficient amount of each commodity. Moreover, for k e K and e e C, denotes a utilization factor for commodity k on edge e. Thus, the first term in the right-hand side of (14) is the total "load" on element e. On e we can install any (integral) number of systems of type i, chosen from among the set $M(e)$; system type i has throughput m_i. Finally, e in C has existing (prior) capacity $A(e)$. Thus equation (14) states that we install enough capacity on e so as to handle its load.

This system is a bit more general than we need. For example, if C is made up of logical links only, then we can expect to have a small number of different link "types" (for example, between two core switches) and any two links of the same type will have identical sets $M(e)$. In an extreme case, every edge has the same set. But we note that the modularities frequently are not integer multiples of one another. In summary, then, we expect to see a relatively small number of different types of integer variables (and, as a result, of the numbers m_i). In any case, formulations of this type are quite "rich" (read, complex) and we do not have anything close to a complete characterization of the polyhedral structure of such a mixed-integer set. Our approach simply seeks to cut-off the very weak fractional approximations produced by the initial LP-relaxations.

By multiplying each inequality (15) by an appropriate nonnegative value m $_k$, and adding all of the resulting inequalities, and all the inequalities (14), one obtains an inequality with the general form

$$\sum_{e \in C} \left(\sum_{i \in M(e)} m_i z_i \right) + \sum_{e \in C} \left(\sum_{k \in K} \alpha_e^k f_e^k \right) \geq R \tag{17}$$

where the a are parameters, and, by choosing the multipliers m $_k$ appropriately, R > 0. To this inequality we can now apply the MIR procedure, and typically there are several possible ways to do so. One way is to choose a particular value m_i in the sum above, say $m_i = B$. If r is the fractional part of R/B, one obtains a valid inequality of the form

$$\sum_{e \in C} \left(\sum_{i \in M(e)} \lceil m_i / B \rceil z_i \right) + \sum_{e \in C} \left(\sum_{k \in K} \left(max\{\alpha_e^k / (rB), 0\} \right) f_e^k \right) \geq \lceil R/B \rceil \tag{18}$$

In order to apply this last rounding step one must choose the index i with $m_i = B$. Frequently (but not always) it may be the case that for all elements of the cutset C there is a common facility type available for use on and which in addition has smallest cost per unit capacity . When this occurs the LP-relaxation will give largest weight to this facility type (possibly zero for all others), and, in fact, just enough weight to handle the load on each particular element

pose integer programming code will fail (abysmally) in all but the easiest problem instances. On the other hand, it is clear that developing any special-purpose algorithm that is even moderately effective will require a great deal of time and effort, and the result will be a large and complex piece of code -- so we want our algorithm to be somewhat "general". In fact, it is to be anticipated that during development of the algorithm and afterward, some of the details of the underlying formulation will change -- this is a natural consequence of the complexity of the underlying "real" problem. Furthermore, it would be desirable that the algorithm be flexible enough that it can be applied to problems other than "pure" ATM network layout problems -- in other words, we want to leverage the large effort required to develop the algorithm and code. Part of the reason for using an algorithm based on linear programming and branching heuristics is precisely the robustness of those two tools -- we just want to balance performance with a long shelf-life for the code. It is worth noting that the constraints of the network layout problem we are dealing with are so intricate that just finding a feasible solution, or testing whether a solution is truly feasible, can be a fairly complex task. Thus, linear programming is not only of use as a solution tool, but as a language to describe the problem.

Our algorithm is made up of three parts: (a) a preprocessing stage, mostly consisting of coefficient tightening and, for extremely large problem instances, a set-covering heuristic used to obtain a restricted family of paths, which we describe later, (b) a cutting-plane stage, and (c) a branching and rounding stage.

4.1 The cutting-plane stage

The above formulation is generally quite weak, and a heuristic that is based on directly rounding the solution provided by the LP-relaxation may well produce solutions of very poor quality. It is essential to strengthen the formulation with appropriate inequalities. Research on network design algorithms has shown, repeatedly, that "cutset" inequalities can be fairly effective. See, for example, [16]. These inequalities consider a cutset in the underlying network (either a node- or edge-cutset) and then make use of the fact that the capacity installed on the cutset has to be large enough to handle the total demand across the cut. Since the capacity variables are integral, an application of the Gomory or MIR procedures [19] yields strong inequalities that generally cut-off the continuous solution. In the simplest possible scenario, we start with an inequality of the form

$$\sum_i m x_i \geq D \qquad (12)$$

where the x variables are capacity allocations for the elements (nodes or edges) that make up a cutset; m is the "modularity" of the (single) transmission system, and D is the total demand "across" the cut. This inequality will be statisfied by the solution to the LP relaxation, but it can be strengthened to

$$\sum_i x_i \geq \left\lceil \frac{D}{m} \right\rceil \qquad (13)$$

which will most likely be violated. Several of the references listed above describe algorithms for network design that are almost single-handedly based on cutset inequalities, and which yield good computational results. Our first task was to apply this approach to our problem. Next we describe two of the ways that we applied this strategy.

$$Minimize \sum_{a \in A} y_a F^a + \sum_{l \in L} \sum_{t \in T} w_l^t H^t \tag{7}$$

$$\text{Subject to: } \sum_{p \in P_k^s} x_p^s \geq d_k^s \qquad \forall k \in K \tag{8}$$

$$\sum_{s \in S} \left\lceil \left(\sum_{p \in Q_a} x_p^s f_p(s) \right) / M_s \right\rceil \leq C^a y_a + C_a \qquad \forall a \in A \tag{9}$$

$$\sum_{s \in S} g_l \left(\sum_{p \in R_l} x_p^s \right) \leq E_l + \sum_{t \in T} w_l^t B_t \qquad \forall l \in L \tag{10}$$

$$y_a \in Z, \forall a \in A; x_p^s \in Z, \forall s \in S, \forall p \in P; w_l^t \in Z, \forall l \in L, \forall t \in T \tag{11}$$

where Z is the set of non-negative integers. Each constraint set (8)-(5) identifies which class of requirement is to be met: (8) for demands, (9) for switch capacity, (10) for link capacity and (11) for integrality. If there are reliability requirements, then it is necessary to add constraints that ensure a minimum level of flow of each commodity-service pair in cases of single or multiple switch node or link failures. (Representation of the latter requires a mapping between the physical network links and logical links, foregone in this exposition for simplicity.) These requirements generally resemble the constraints set (8)--(10) with one or more nodes excluded for each failure scenario. The implementation reported here does incorporate these reliability constraints.

4. SOLUTION PROCEDURE AND HEURISTICS

The mixed-integer program described in the previous section presents some unique challenges. Network design problems have been extensively studied in the literature (see, for example, [9],[10],[11], [12],[13],[15],[16],[18] and references therein) with good experimental results using cut-and-branch (see [14] for mathematical programming basics). However, traditional models tend to stress *one* particular feature, typically link transmission systems (modeled as edge capacities in an underlying network), or, less frequently, switch capacity (arising as node capacity). Typically, the models allow arbitrary paths to be used for the routing, or perhaps a (simple) restricted family of paths, such as limited-length paths. See [8] for a more comprehensive recent model.

What makes the above problem difficult is (a) the large number of different, but interrelated, capacity investment decisions to be made (switches, plug-ins, link transmission systems, concentrators, etc.) which in addition have unpredictable cost profiles, making it impossible to simplify the model in favor of one with fewer integer variables, (b) the complicated nature of the path family that we use, together with the complex way that the paths use the various capacities, and (c) the potentially very large size of the formulation. Often, the problem has a very large duality gap, and gives rise to large, difficult and ill-conditioned linear programs.

When devising an effective algorithm for handling problems of this sort it might be best to exercise some careful “strategic” planning. From experience, we know that a general-pur-

deal with details of this array, but we note that it is used to determine the gain function defined below.

Let K be the set of all node-pairs (or commodities) with a demand between them. A *virtual path* between a node-pair is simply a path in the *logical* network G between the two nodes. Virtual paths may be pre-selected for their appropriateness for carrying a given service, based on delay requirements of the service, maximum cell loss, or other appropriate criteria. Assuming that service-based pre-selection occurs, there will be a set of paths associated with each node-pair and service. For service s and commodity k, $P_k{}^s$ denotes the associated set of virtual paths and d_k^s is the required *flow* of service type s for commodity k. We let P denote the set of all paths, i.e., $P=U_{k,s}P_k{}^s$. Finally, let Q_a be the set of all paths in P that visit the switching node a and let R_l be the set of all paths in P that contain the link l.

The spare capacity of an existing switch at node a is given by C^a, and the spare transmission capacity on link l is defined to be E_l. We let F^a and C_a denote the cost and capacity of each additional switch at location a. Thus, the values of $k_A{}^a$ and C_a are determined by the type of switching units that may be placed at a node a --in our studies either backbone or edge switches. We let T be the set of available transmission facility types. For transmission facility t in T, B_t denotes its capacity and H^t its cost.

We let $G_l(\underline{x})$ denote the scalar *gain* of an aggregate set of multiplexed flows for a vector x which is a mix of demand flows $\underline{x} = \left(x^1, x^2, \ldots, x^S\right)$, where x^s is the *number* of *concurrent sessions* of service type s multiplexed in the link l. As pointed out in Section 2, parameters such as the buffer capacity of each switch port that affect the gain function are implicitly incorporated via index l, terminating on one or two switch nodes, on which G_l is explicitly dependent. The gain function is typically concave, at least with respect to each x^s, with $G_l(\underline{0}) = 0$ for all l in L. It takes into account the traffic characteristics of the service and the switch's ability to multiplex various circuits of services into a trunk size smaller than the linear sum of the peak bandwidths. For the purposes of this paper we assume that multiplexing is only realized within each service type and for aggregation of different services, the linear sum of their multiplexed rates is used. In the terminology of Section 2

$$G_l(\underline{x}) = \sum_{s=1}^{S} g\left(x^s, B, \varepsilon_i\right) \tag{6}$$

Closely associated with the gain function is the function $f_p(s)$ which determines the port utilization of the service s as carried on path type p when it enters a switching node a (dependence on a is implicit via p). This function yields a value of twice the peak rate of a flow $x_p{}^s$ for service s on path p if p enters a switch from an end node and exits towards another end node. Alternatively, it could be equal to the gain function of the total flow if the path traverses an inter-switch trunk. Thus, the value of the function depends on the characteristics of the related path. Let M_s represent the modularity of service s on the switch .

Finally, the decision variables for the model are $x_p{}^s$, the number of sessions of service type s assigned to virtual path p, y_a the number of switches added at candidate switch location a, and $w_l{}^t$, the number of transport systems of type t added to logical link l. With the above notation, the minimum cost ATM network design problem may be described by:

$$Pr(Q > B) \approx \left(\frac{2^{1-\frac{1}{H}} D_0^{\frac{1}{H}-2}}{\sqrt{H - H^2}} \right) \lambda e^{-\frac{D_0^2}{2}} \tag{3}$$

where

$$D_0 = \frac{B^{1-H}(C-m)^H}{H^H(1-H)^{1-H}} \tag{4}$$

and λ is a constant ($\lambda \approx 1.4 - H$). Thus, to meet a pre-specified quality of service -- QoS -- either in terms of cell loss ($Pr\{Q_i((t) > B)\}$) or delay, the output rate of the buffer C can be calculated from inversion of (*). Similarly, for N flows of the same service, we invert another equation of type (*) for an fBm with the mean rate $M_i = Nm_i$, $A_i = Na_i$ and same H_i. We shall refer to this last inversion as the *gain* function, $g(N_i, B, \varepsilon_i)$, which gives the minimum output rate needed to support N_i flows or *sessions* of service i for a given cell-loss QoS parameter ε and buffer capacity B cells. If in addition to the fBm description of each service, each service i also has a practical upper limit on its cell rate, say Π_i, then the ratio

$$\frac{N_i \Pi_i}{g(N_i, B, \varepsilon_i)} \tag{5}$$

is known as the *multiplexing gain* of homogeneous mix of service i. It is possible to allow for a *heterogeneous* mixing and multiplexing of different services on a given link and attempt to determine the minimum rate needed to support each service at it's own QoS threshold in the mix. Although it's possible to approximate such a mix via more generalized models of processes, for this paper we assume that output rate for the superimposition process is given by the linear sum of its individual gain functions for each service type. This approach is certain to give a conservative estimate of rate needed for the multiplexing of heterogeneous services.

3. NOTATION AND PROBLEM FORMULATION

In this section we describe a simplification of the optimization model addressed by the network design engine. Our goal is to give the flavor of the underlying optimization problem without going into full detail. The network underlying the design problem is modeled as a graph *G=(N,L)* where N is the set of demand endpoints and switching nodes and L is the set of logical links connecting them. A logical link either connects a demand endpoint to a switching node (access link) or connects switching nodes to each other (trunks). We assume that the switching nodes form a subset A of N. These are the locations where backbone and edge switches may be deployed. We note that G is the logical network derived from an underlying physical network.

A data network may carry demands for several different services. We let S denote the set of service types available. The cardinality of this set is denoted by $|S|$. Associated with each service s in S is a traffic descriptor array that provides such parameters as the mean rate of the service, the peak rate of the service, second or higher order statistics of its bit rate, and possibly other descriptors of interest such as its peakedness and correlations. We will not

It's possible to write down the necessary constraints that characterize requirements one to six in the above list and cast the problem as one of optimizing the seventh, namely, the total additional cost of switching plus that of transport. The resulting problem then is not only a multicommodity network flow with non-linear side constraints, but also integrality and modularity of network components need to be accounted for in addition to guaranteed levels of throughput in cases of single physical layer failures. The resulting problem is naturally hard to solve mathematically to optimum.

In this paper we report on some of the techniques that we've successfully used to model fast-packet traffic -- such as fractional Brownian motion (fBm) -- and manage its non-linearities for dimensioning as well as bound the growth of the feasible paths for commodities and solve the multi-scenario minimum cost capacity expansion problem that arises as a result. The optimization-based heuristics used not only have been successfully applied to dense networks of up to 250 nodes, but also result in solutions with relatively small gaps with respect to their (consolidated) linear programming lower-bound relaxations.

The outline of this paper is as follows. Section 1 introduces the traffic models to be used and describes the resulting dimensioning methods. Section 2 provides the notation and formulation of the mathematics programming problem. Section 3 describes the disaggregation of the problem into simpler optimization components that form a heuristic solution to the original problem. Section 4 provides data used for sample runs of the optimization code. Finally, Section 5 provides computational results and presents a discussion of the economic effects of multiplexing in ATM networks based on the computational experiments.

2. TRAFFIC MODEL

The traffic model we use, fractional Brownian motion, fBm, was introduced by Norros [1] based on the general characteristics of high-resolution fast packet data traffic traces collected and analyzed by Bellcore researchers [2]. We therefore assume each *service flow* pumps out ATM cells roughly according to a fBm source S_i, and thus at time t the total flow is equal to

$$A_i(t) = m_i t + \sqrt{a_i m_i} Z^H(t) \tag{1}$$

where m is the cell rate, a is the variance of the cell rate per unit of time, and $Z^H(t)$ is the standard fBm (zero mean, unit variance with $Z^H(0) = 0$) with Hurst parameter H. The Hurst parameter gives the index of the power-law decay of the autocorrelation function for large time lags and is in the range [0.5,1]. For this model as input rate of cells to a switching system with output rate C, the build up of cells at time t, $Q_i(t)$, in the switch's buffer will be according to

$$Q_i(t) = \sup_{s \le t} \{A_i(t) - A_i(s) - C(t-s)\} \tag{2}$$

see [3]). Thus, using above, it is possible to determine $Pr\{Q_i((t) > B)\}$ which is the percentage of cells overflowing the buffer size B and thus lost . It is shown in [4] that for large enough B the latter probability is equal to

ATM network design:
Traffic models and optimization-based heuristics

Iraj Saniee†, Daniel Bienstock‡

†BELLCORE
Morristown, NJ 07960
iis@bellcore.com

&

‡Dept. of IEOR
Columbia University
New York, NY 10027
dano@ieor.columbia.edu

Abstract

We consider the design and capacity expansion of ATM networks as an optimization problem in which flows representing end-to-end variable bit-rate services of different classes are to be multiplexed and routed over ATM trunks and switches so as to minimize the costs of additional switches and transport pipes while meeting service quality and survivability constraints. After discussing the underlying fractional Brownian motion models for flows, a non-linear multicommodity optimization problem is formulated and heuristics for its approximate solutions are described. Finally, computational results are produced that demonstrate realistic size problems can be solved with the proposed method to shed light on key economic characteristics of ATM traffic, such as safe levels of statistical multiplexing, as well as robust and efficient design alternatives.

1. INTRODUCTION

ATM switches have begun to be deployed in the backbone networks of a variety of fast packet services such as Frame Relay and IP. The efficiency of ATM due to its ability to integrate a variety of services is thus becoming a reality. However, to take full advantage of its multi-service capabilities, it's important that ATM networks be designed with a variety of desired characteristics in mind.

First, fast packet data traffic need to be understood so that appropriate dimensioning can be performed to balance quality of service against overprovisioning. Second, it's necessary that there be intra-service multiplexing gains to make ATM competitive with efficiencies existing in single service networks. Third, to the extent that it's understood, inter-service multiplexing need to be allowed for, or in case of service priorities, guarded against. Fourth, a level of flow-based restoration at the traffic layer needs to be built in. Fifth, appropriate virtual connections need to be set-up to simplify routing. Sixth, primary and restoration routings that meet service constraints need to established that best take advantage of existing or planned bandwidth. Seventh, a design needs to be established that adequately balances switching costs against those of transport.