Joint Planning of the Physical and Logical Configuration of ATM Networks

András Faragó    Tibor Cinkler    Vu Thanh Hai    Szabolcs Malomsoky
Dept. of Telecommunications and Telematics
Technical University of Budapest
XI. Stoczek u. 2, Budapest, Hungary H-1111
E-mail: farago@ttt-atm.ttt.bme.hu

Abstract

An ATM network planning model is presented, along with algorithmic considerations. The procedure results in a physical network topology with a virtual path layout and transmission capacities assigned to the VPs. Thus, the physical and logical configuration design tasks are integrated into a single optimization procedure. Predetermined connectivity requirements are satisfied as constraints, and the objective is to minimize an overall cost function. We also consider another case when the physical network is already given and only the VP layout has to be planned.

1 Introduction

ATM network planning is a subject that has attracted relatively little attention in the literature so far. Most papers deal with the dimensioning and logical configuration aspects, rather than designing the physical network topology itself. Even in the latest Networks conference there was not even a single paper focused specifically on ATM network planning as such, that is, to the design of the physical topology. The papers in the ATM Network Planning Session addressed the logical configuration issues, assuming a given physical network, see e.g. [1]; [2] (a further developed version of the latter was published as a journal paper [3]).

The potential reason for this lack of exploration could be that ATM is not expected to stand alone as a generic physical network. Rather than that, a large public ATM network will typically be carried by SDH or SONET high speed synchronous networks. This may result in the view that the network planning issues of ATM are shifted to SDH (SONET) planning. Thus, when it comes to ATM, then only the logical configuration problems are investigated, while the physical carrier network is assumed given.

The above view, however, misses the point that the design of the carrier infrastructure network cannot be independent of the carried ATM network, at least in case one aims at a good cost/performance ratio and high utilization of resources. On the other hand, it seems not at all easy to find an ATM network planning model that captures the most relevant points, but it still avoids the overcomplications and the danger of intractability caused by the simultaneous handling of the infrastructure and the carried network.

In the present paper we attempt to develop an ATM network planning model and an algorithmic solution. Similarly to most practically relevant cases of network design, finding the global optimum is algorithmically intractable for large networks, as computationally hard (NP-complete) problems are involved. Therefore, we present a procedure that approaches a local optimum of the cost function. It is demonstrated numerically that the method produces good results.

Once the physical network is built and fixed, the original procedure cannot be applied again if only the logical configuration has to be changed for changing VP demands. This is why we also consider the problem of finding a feasible VP layout for given VP capacities in a given physical network. This is another difficult algorithmic problem for which we propose an approximative solution.

The motivation and intended first application of these and related methods is to implement them in an intelligent network management support tool, called PLASMA [4]. The name PLASMA stands for PLAnning and Simulation methods in network MANagement. The system development started at the High Speed Networks Laboratory of the Department of Telecommunications and Telematics, Technical University of Budapest, in cooperation with Ericsson and is continued now in Traffichab, a competence center of Ericsson, established in Budapest Hungary. This software tool is capable to support the work of the network traffic manager by offering and testing optimized network configuration versions, depending on the actual...
traffic situation, demands and many other conditions. Because of the flexible logical (re-)configuration possibilities of modern high speed networks, there are quite a few tasks for the network manager that can be solved by such methods that are closer in spirit to planning tasks. This foreseen convergence of a substantial part of the design and management methodology makes it relevant to implement planning methods in the management support tool PLASMA.

2 The Model

We propose the following model for the considered ATM network planning task. There are $N$ network nodes, numbered by $1, 2, \ldots, N$. For each pair $(p, q)$ of nodes a transmission capacity demand $T_{pq} \geq 0$ is given, expressed as an integer multiple of an elementary unit. Note that from the teletraffic viewpoint $T_{pq}$ is not the same as the offered traffic: we assume that the offered traffic has been already mapped into capacity demands, using available teletraffic dimensioning methods. Thus, $T_{pq} \geq 0$ is the transmission capacity to be engineered to carry the traffic between $p$ and $q$. (For simplicity we assume symmetry in the sense of $T_{pq} = T_{qp}$, but the model can directly be extended to the non-symmetric case, as well).

An essential feature of ATM is that the traffic in the ATM network is sent through virtual paths (VPs). Network safety and load balancing considerations require that the same pair of nodes (origin-destination pair, O-D pair for short) is to be connected generally by several independent virtual paths that have common end-nodes but do not share internal nodes. To describe these requirements, for each O-D pair $(p, q)$ of nodes a number $k_{pq}$ is assumed given. The quantity $k_{pq}$ prescribes the number of needed independent VPs that connect $p$ and $q$. Again we assume symmetry in the sense of $k_{pq} = k_{qp}$, for simplicity.

The cost structure of the network is modeled as follows. All costs, as often done in network design, are mapped onto the links. The global cost is the sum of individual link costs. The cost $c_{ij}$ of a link $(i, j)$ that connects nodes $i$ and $j$ has the following structure:

$$c_{ij} = a_{ij} + f_{ij}(x_{ij}).$$

Here $a_{ij}$ is the fixed cost of the link that is associated with creating the link. The variable cost $f_{ij}(x_{ij})$ is a function of the transmission capacity $x_{ij}$ that is “put on the link” during the planning procedure. Our method allows an arbitrary variable cost function. For the numerical experiments we chose linear functions of the form $f_{ij}(x_{ij}) = b_{ij}x_{ij}$, which resulted in link costs of the form $c_{ij} = a_{ij} + b_{ij}x_{ij}$.

The aggregated global cost is the sum of the individual link costs:

$$G = \sum c_{ij} = \sum (a_{ij} + f_{ij}(x_{ij})).$$

where the summation is taken over those links that are created by the planning algorithm and $x_{ij}$ is the summed capacity of virtual paths that traverse the link $(i, j)$. Note that the VP routes and capacities are not given in advance, they are also to be obtained by the algorithm.

Thus, the algorithmic task can be formulated as follows.

**Given:**

- $N$: number of nodes
- $T = [T_{pq}]$: capacity demands
- $K = [k_{pq}]$: VP connectivity demands
- $a_{ij} = a_{ij} + f_{ij}(x_{ij})$: link cost functions

**Find:**

- Which physical links are to be built
- Transmission capacity of each physical link
- $k_{pq}$ independent VP routes to connect each O-D pair $(p, q)$
- Transmission capacity of each VP

**Subject to:**

- The physical capacity of any link cannot be smaller than the summed capacity of the VPs that use the link.
- The summed capacity of the VPs that connect the same O-D pair is at least as large as the demand for the O-D pair.

**Objective:** The aggregated global cost of the created physical links is as small as possible.

It is not difficult to see that the problem belongs to the family of hard (so called NP-hard [5]) combinatorial optimization problems, as it contains the well known Traveling Salesman problem [5] as a subcase. To see this it is enough to take all link costs equal to the fixed part of the cost, while the variable part is set identically to 0. If we also set $k_{pq} = 2$ for all $(p, q)$, then it is easy to see that the optimal solution is a minimum cost traveling salesman tour.
3 Algorithmic Solution

In this section an algorithmic solution is presented for the above specified ATM network planning task. First we show how to obtain an optimal system of \( k_{pq} \) independent virtual paths between a single given O-D pair \((p, q)\), such that the capacity demand (load) \( T_{pq} \) is distributed equally among the \( k_{pq} \) VPs. Optimality means here that we want to create the new virtual paths with the smallest possible increase in the global cost of the existing network. That is, we want to extend the network to the smallest possible extent (in the cost sense) to satisfy the demand for a new O-D pair. Then the solution of this simpler problem is used as a subroutine for solving the main task.

3.1 VP Routing Between a Single O-D Pair for Minimum Cost Network Extension

Let us assume we already have an existing physical network in which the links have certain capacities. Let \( R_{ij} \) be the capacity of link \((i, j)\). We wish to find \( k_{pq} \) independent virtual paths between a given O-D pair \((p, q)\), such that the load (demand) \( T_{pq} \) is distributed uniformly among the VPs and the additional cost of creating the new VPs is minimum. It is assumed that the existing link capacities are already used by existing VPs because the design procedure has not put unnecessary excess capacity on any link in earlier phases.

We find the required minimum cost extension by reducing the task to a minimum cost network flow problem, as follows.

**Algorithm Minimum Cost Extension (MCE)**

**Step 1:** Define the instance of a minimum cost flow problem as follows.

1. The underlying graph is a complete graph on \( N \) nodes.
2. Assign unit capacity to each edge.
3. If the physical link \((i, j)\) exists in the network, then assign the cost \( c_{ij} \) to edge \((i, j)\)

\[
\bar{c}_{ij} := a_{ij} + f \left( \frac{T_{pq}}{k_{pq}} \right).
\]

4. If \((i, j)\) is a non-existing physical link in the network then assign the cost to edge \((i, j)\) by

\[
\bar{c}_{ij} := a_{ij} + f \left( \frac{T_{pq}}{k_{pq}} \right).
\]

5. Assign a flow demand of \( k_{pq} \) to the O-D pair \((p, q)\).

**Step 2:** Find a minimum cost integer flow in the flow problem specified in Step 1. (This can be done by existing network flow algorithms, see e.g. [6]).

**Step 3:** Due to the specially defined instance of Step 1, the minimum cost integer flow will consist of \( k_{pq} \) unit-flow path between \( p \) and \( q \). These paths are the required VPs, each with capacity \( T_{pq}/k_{pq} \).

The special cost assignment \( \bar{c}_{ij} \) ensures here that only the additional cost is incurred. If \((i, j)\) is an existing link, then \( f(R_{ij} + T_{pq}/k_{pq}) - f(R_{ij}) \) is the cost increase if it is used by a VP of capacity \( T_{pq}/k_{pq} \). If the link has not yet been built, then the cost of using it contains the fixed cost, as well, this is represented by \( \bar{c}_{ij} := a_{ij} + f(T_{pq}/k_{pq}) \). Note that the procedure is not sensitive to the choice of the functions \( f(\cdot) \), it works for arbitrary functions.

It is well known from the theory of network flows (see e.g. [6]) that for integer demand and integer edge capacities the demand can be satisfied by an integer flow (if there is a solution at all). The existence of a solution is guaranteed here by the fact that we are allowed to build any new links if necessary. (Of course, as a trivial condition, it is assumed that \( k_{pq} \leq N - 1 \), otherwise it would be impossible to have \( k_{pq} \) independent VP routes). Thus, we see that Algorithm MCE really solves the subtask we addressed in this subsection. The solution is efficient in the sense that it runs in polynomial time (the complexity does not grow exponentially with the size of the network). This follows from the same property of network flow algorithms. Since we run a minimum cost flow algorithm for each O-D pair, therefore, the complexity is \( O(N^2F) \), where \( F \) is the complexity of the minimum cost computation. The construction of the network flow instance in Step 1 takes \( O(N^2) \) time, so it does not increase the \( O(N^2F) \) term.

3.2 The Network Planning Algorithm

In this subsection we show how to use the MCE procedure, presented in the previous subsection, to solve the network design task. The idea is that starting with an empty network (containing no links), VPs
are added step by step for each O-D pair, making a minimum cost extension of the network in each phase by the MCE algorithm.

**Algorithm VP-Based Network Planning (VPNP)**

**Step 0:** Initialization: Take an empty network (no links)

**Step 1:** Run the MCE algorithm with the present network, separately for each O-D pair that has not yet been considered.

**Step 2:** Choose an O-D pair \((p,q)\) that produced the smallest extension cost among the O-D pairs tried in **Step 1**.

**Step 3:** Do the extension with the minimum extension cost O-D pair found in **Step 2**.

**Step 4:** If all O-D pairs with \(T_{pq} > 0\) have been connected by VPs then STOP else go to **Step 1**.

It is clear from the construction that this algorithm has polynomial running time because it essentially consists of \(N + (N - 1) + (N - 2) + \ldots + 1 = N(N + 1)/2 = O(N^2)\) calls of the MCE algorithm. This yields an overall complexity of \(O(N^4F)\) where \(F\) is the complexity of a minimum cost computation.

To further decrease the overall cost, we can iterate the VPNP algorithm, as follows. The idea is that we can potentially improve the VP system connecting a given O-D pair if we remove these VPs from the network and recompute them again using the MCE algorithm. Note that the new result is not necessarily identical with the previous one, since the network has changed when other VPs were added for other O-D pairs. On the other hand, the possibility of putting back the previous VP system for the O-D pair is always open. This implies that the recomputation procedure cannot degrade the already existing solution. The details are omitted here for space limitations, the interested reader is referred to [7].

**4 A numerical example**

We demonstrate the algorithm on a network of 15 nodes. The nodes are randomly placed on the plane and the link fixed cost is proportional to the distance of the end nodes of the link. The variable cost is linear and it is also proportional to the distance. Let \(a\) be the cost of the unit capacity over unit distance. Then the link cost in this case can be expressed as

\[
e_{ij} = (1 + ax_{ij})d_{ij},
\]

where \(d_{ij}\) is the distance between nodes \(i, j\) and \(x_{ij}\) is the transmission capacity “put on the link” by the VPs that traverse it.

The value of the parameter \(a\) can control the ratio of the fixed and variable cost. To make the numerical example easy to overview, we took \(T_{pq} = 2\) and \(x_{pq} = 2\) for each O-D pair \((p,q)\). That is, we look for a 2-connected network in which the capacity of each VP is 1, measured in appropriate relative units.

The resulting networks for \(a = 0.005\) and \(a = 0.015\) are shown in Figures 1 and 2, respectively. It is clearly seen that the two networks are different. If \(a\) gets larger, then it is worth building more links because the loss caused by building more links is dominated by the gain that shorter VP routes are possible, which reduces variable costs, since the capacity is put on less links. It is interesting to note that the network of Fig. 2 is not a simple extension of the one in Fig. 1, the denser network does not arise by just adding new links to the network of Fig. 1.

For comparison we have searched for a minimum cost ring network on the 15 nodes, as well. The ring topology is extremal in the sense that it can ensure 2 VPs between each O-D pair such that the total number of links is minimal. Also, the VP layout is unique in the ring, since there are only two possibilities to connect any two nodes (clockwise and counter-clockwise). Finding the minimum cost ring is, however, still difficult from the algorithmic viewpoint because it is essentially equivalent to the Traveling Salesman problem. Nevertheless, for a 15-node network we could still solve it in reasonable time by simulated annealing. The resulting networks coincide for \(a = 0.005\) and \(a = 0.015\). The obtained ring network is shown in Fig. 3.

Comparing the costs of the networks obtained by our algorithm to the reference optimal ring network, we find the following effect. If \(a\) is small (0.005 in our case) then the cost of the ring (Fig. 3) and the cost of the network found by our algorithm agree with less than 1% difference, although the two networks are different. If \(a\) gets three times larger (0.015), then our algorithm provides a solution with 15.6% smaller cost than the optimal ring. This clearly shows that if the capacity dependent part of the cost has higher relative weight, then it is worth deviating from the ring, which is the simplest 2-connected solution, because with “chord-links” we can achieve substantial savings in cost.

Fig. 1  
Fig. 2  
Fig. 3
5 VP Layout Design in a Given Physical Network

The procedure presented in the previous section designs the physical and logical configuration in an integrated way. It is expected to happen, however, that having built the physical network, the demands for the logical configuration may change. For this case we present a new algorithm to find a feasible VP layout with given VP capacity demands in a *given* physical network. For space limitations, the procedure is presented for the basic case when at most one VP is required between each O-D pair; this can be extended to the case with VP protection, as well. Proofs are omitted in this short conference version, they will be presented in the full journal version.

Let $V_j$ be the given capacity demand of the $j$th VP and $C_j$ be the physical capacity of link $j$. A feasible solution of the problem is a set of VP routes such that on each link $j$ the sum of the $V_j$ values for those VPs that traverse the link does not exceed $C_j$. Deciding whether a feasible solution exist at all is a difficult (NP-complete) combinatorial optimization problem.

On the other hand, not all feasible solutions are equally good from the practical viewpoint. If a route system saturates or closely saturates some links, then it is not preferable because it is close to being overloaded. For this reason, let us assign a parameter $0 < \rho_j < 1$ to each link $j$, such that $\rho_j$ will act as a "safety margin" for the link. More precisely, let us call a feasible solution a safe solution with parameters $\rho_j$ if it uses at most $\tilde{C}_j = \rho_j C_j$ capacity on any link $j$.

Now, the interesting thing is that if we restrict ourselves to safe solutions, then the hard algorithmic problem becomes solvable by a nontrivial but efficient algorithm! In other words, it can be proven that if a safe solution exists, then we can efficiently find it. The price for this is that we "cut down" a small fraction of the solution space: we exclude those solutions that are originally feasible, but not safe.

Now let us choose the safety margin $\rho_j$ in the following way. If the network has $m$ links, then set

$$\rho_j = 1 - \sqrt{\frac{\ln 2m}{C_j}}.$$ (1)

Note that this approaches 1 if $C_j$ grows, even if the network also grows but $C_j$ grows faster than the logarithm of the network size, which is reasonable (note that doubling the network size will increase the logarithm by less than 1). For example, if in a network each link capacity is 1,000 units (measured in relative units, such that the maximum call bandwidth is 1) and the network has 200 links, then $\rho \approx 0.98$.

Now we briefly outline how the algorithm works.

**Algorithm**

*Step 1 Initialization*

Compute the $\tilde{C}_j = \rho_j C_j$ values with $\rho_j$ set according to (1).

*Step 2 Flow relaxation*

Solve the continuous minimum cost multi-commodity flow relaxation of the problem via standard linear programming, using the $\tilde{C}_j$ capacities. (Note that although this finds a flow of the required value between each O-D pair, nevertheless, it does not yet provide VP routes, since the flow typically branches arbitrarily into small parts rather than going on one path, this is why it is called a relaxation of the problem). In case this flow problem has no solution then declare "no safe solution exists" and STOP.

*Step 3 Integralization via Random Walk*

For each O-D pair $u_i, v_i$ find a VP route as follows. Start at the origin and take the next node such that it is drawn randomly among the neighbors of the origin, with probabilities proportional to the $i$th commodity flow values on the links from $u_i$ to the neighbors (a directed graph is assumed). Continue this in a similar way: at each node choose the next from its neighbors as above. (Note that a circle cannot arise on the way, because that would mean a circle with all positive flow values, which could be cancelled from the flow, thus contradicting to the minimum cost flow). Finally we have to arrive to $v_i$ and we store the found $(u_i, v_i)$ route.

*Step 4 Feasibility Check and Repetition*

Having found a system of routes in the previous steps, check whether it is a safe feasible solution. If so, then STOP, else repeat from *Step 2*. If after repeating $r$ times ($r$ is a fixed parameter) none of the runs are successful then declare "No safe solution is found" and STOP.

It is clear from the above brief description that the algorithm has practically feasible complexity, since essentially the most complex part of it is linear programming. It is repeated $r$ times where $r$ is a parameter, chosen by us. The role of $r$ and the main property of the algorithm is show in the following theorem.

**Theorem:** With probability at least $1 - 2^{-r}$ the algorithm either finds a safe solution of the VP layout problem or verifies that none exists. The third
outcome, when a safe solution possibly exists but the algorithm could not find it, can happen only with probability smaller than $2^{-r}$.

Thus, if e.g. $r = 100$, then the probability of missing an existing solution is smaller than $2^{-100} \approx 10^{-30}$, which is practically zero.

6 Acknowledgment

The work has been done in the framework of a research cooperation between Ericsson and the High Speed Networks Laboratory at the Department of Telecommunications and Telematics, Technical University of Budapest. The authors are grateful to Miklós Boda, Géza Gordos and Tamás Henk for their continuous support and encouragement.

References


