Balancing VP and VC Routing in ATM Networks

Åke Arvidsson
University of Karlskrona/Ronneby
Dept. of Telecommun. and Maths., Univ. of Karlskrona/Ronneby, S-371 79 Karlskrona, Sweden. Email: akear@itm.hk-r.se
Tel: +46 455 78053. Fax: +46 455 78057.

Abstract

Virtual paths (VPs) facilitate rapid movement of end-to-end traffic streams in ATM networks by keeping processing at intermediate nodes at a minimum. For low volumes of traffic, however, VPs may suffer from poor statistical multiplexing gain, and thus result in low utilisation of transmission resources. Balancing between processing and utilisation, we study a procedure whereby lightly loaded VPs are decomposed into at most two segments, which require only one intermediate switching for an end-to-end traffic stream. The resulting segments are then aggregated with other segments or existing VPs. The resulting VPs can thus achieve higher multiplexing efficiency for marginally increased processing. For our test networks, experimental results show that about 95% of the lightly loaded VPs will be decomposed and aggregated by our procedure. When applied to networks where the VPs are adjusted over time to meet traffic variations, the procedure results in an increase in carried traffic and reduced overall overhead by providing a more stable VP network configuration with fewer modifications.

Keywords

Virtual path network, Virtual path routing, Virtual channel routing.

1 PRELIMINARIES

1.1 Definition of Virtual Paths

The virtual path (VP) and virtual path connection (VPC) are important concepts in ATM networks. A VP recognises the distinct identity of a traffic stream between two nodes. Cells belonging to a particular VP are identified by a common, fix VP identifier (VPI). A VPC consists of a series of concatenated VPs (possibly with different VPIs) and specifies a route to be traversed by a traffic stream from an originating node through a number of intermediate

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35353-1_28

D. Kouvatsos (ed.), *Performance Analysis of ATM Networks*

nodes to a terminating node. Using VPCs allows for faster set-up of new connections (along predefined routes) and rapid movement of traffic (ATM cells) with minimal processing at intermediate nodes (which is related to switching costs at each node).

A VPC may also be assigned a certain bandwidth, *i.e.* a certain part of the bandwidth on each of the transmission links of the VPC may be reserved for its exclusive use. This accelerates the set-up of new connections (since exclusive ownership of bandwidth means that the availability of resources can be determined over the entire VPC).

A further simplification and speed up is obtained if service classes are segregated into logically distinct VPCs such that each service class has its own logically independent VPC network. With homogeneous traffic, the bandwidth reserved on a VPC may be expressed in terms of the maximum number of simultaneous connections it can support before violating some quality-of-service (QoS) constraints (*e.g.* cell loss). We call this number the "equivalent number of circuits" (or just "circuits" for short) for the traffic type considered, and express VPC bandwidths in terms of these rather than bits per time unit.

We briefly mention that the calculation of the number of circuits from a given bandwidth is a complicated process which is done independently of the work presented here. It involves traffic characteristics (*e.g.* burstiness on the cell and burst scales and QoS constraints), and system properties (*e.g.* buffer capacities and strategies, and service policies).

1.2 Management of Virtual Paths

Because of the number of routes that traffic between a pair of end nodes may traverse, and the number of all possible pairs of end nodes with specified traffic requirements, a frequent use of VPCs may give rise to inordinately large numbers of them. This may cause problems in two areas of ATM network management. One problem area is in the management of VPIs. The maximum number of VPIs is limited by the field length allowed for VPIs in ATM cell headers. The other problem area, which occurs only if VPCs are associated with reserved bandwidths, is in bandwidth management or efficient utilisation of the transmission capacity on transmission links. This is particularly evident for VPCs the traffic of which vary over time (thus not permitting full utilisation during off-peak periods), and VPCs with small volumes of traffic (which do not allow full exploitation of statistical multiplexing).

In a series of works, *e.g.* (Arvidsson 1994, Arvidsson 1995), we have studied automatic reconfiguration of VPCs as a remedy for efficiency problems associated with time varying traffic. The basic concept is to reorganise the network of VPCs with respect to routes and bandwidths over time, in accordance with current demands. We have tried both on-line methods (based on short time traffic demand sampling followed by VPC network redesign) and off-line

methods (where VPC networks are designed according to averages of several demand measurements over the same traffic period). In this paper we extend our work and for the first time address the efficiency problems associated with low traffic volumes on some VPCs.

2 STATEMENT OF PROBLEM

2.1 Virtual Path Efficiency

In network planning, design and management, there often exist competing goals and objectives, for which network planners have to strike a balance and arrive at a happy mean. One such case is ATM VPC routing, where the wish for simplicity in establishing, maintaining, and clearing connections contradicts the wish for high utilisation of the transmission network.

A simple VPC identifies a route for a traffic stream between a pair of end nodes. Simplicity is achieved if traffic streams between pairs of end nodes are carried by end-to-end paths. This is because intermediate nodes mean individual routing of each request in addition to intermediate switching of the traffic flows, both of which incur processing time and require processing logic that translate into additional equipment costs. Moreover, intermediate switching incurs increased cell delay and loss through the buffers involved that again translates to costs in terms of additional equipment requirements. A more advanced VPC may have a certain transmission capacity committed along its route. This simplifies the establishment and clearing of a connection since these can be done end-to-end. The result is less pressure on the processing logic that translates into reduced equipment costs. Finally, a VPC may be devoted to a particular service class in order to simplify call acceptance control (CAC), by means of the equivalent circuit concept, and QoS management, through the distinction of different requirements provided by the VPIs.

Separate VPCs for each service class with dedicated capacity means that VPCs and their committed circuits are treated as a distinct inviolable entities, and we do not allow any statistical sharing of circuits between VPCs. Network management, QoS management and connection management of VPCs and VCCs are indeed simplified this way. On the other hand, resources on transmission links are best utilised when there is maximum sharing among all VPCs traversing common transmission links. We are thus faced with the problem of choosing between reduced management efforts or high link utilisation.

To illustrate the point, consider Figure 1. The diagram to the left shows the number of intermediate switching points per flow *vs.* the size of VPC in terms of number of hops. Though the numbers are not very interesting themselves, it is immediately clear that longer VPCs mean less intermediate switching. (The figures given refer to our specific test networks, which are described

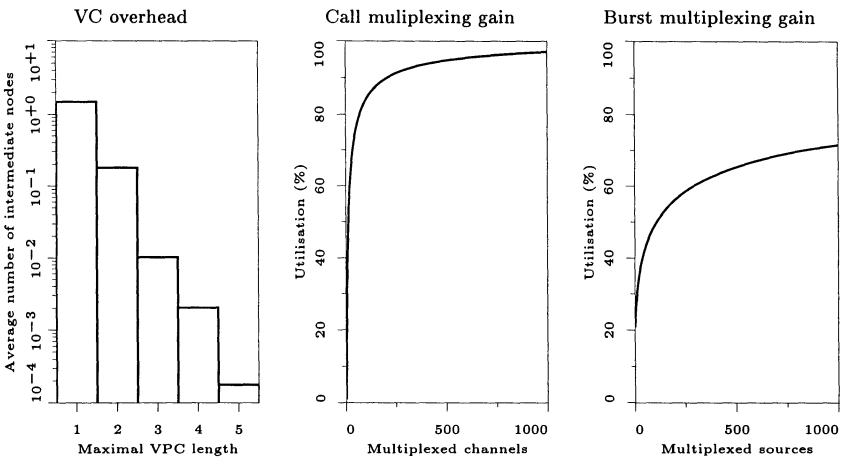

Figure 1 Additional switching points *vs.* VPC length (right). Utilisation efficiency *vs.* traffic volume on the call scale (middle) and burst scale (right).

in more detail below.) The middle and right diagrams show utilisation on the call and burst scale respectively *vs.* multiplexed number of channels and number of sources respectively. Again the numbers are not very interesting themselves, but we notice that the larger the traffic, the higher the utilisation for a constant QoS. It is also seen that there appears to be a limit above which further volumes do not significantly improve the utilisation. (The middle curve is computed from the Erlang-B model with a call loss probability QoS requirement of 10^{-2}. The right curve from the (not so realistic) model of (Anick *et al.* 1982) where independent, statistically identical sources alter between independent, exponentially distributed on- and off-periods and transmit at peak rate during the former where the peak to mean ratio is 10, the buffer size is 10 average bursts, and the cell loss QoS requirement 10^{-8}.)

Comparing the left diagram, which points at decreasing costs for longer dedicated VPCs with fewer intermediate switching points, to the middle and right diagram, which indicate reduced costs for shorter, general VPCs which can achieve high utilisation, it is clear that we are faced with a problem of striking a balance between the two factos. Indeed, as was suggested already by (Burgin 1989), the best choice is a compromise where the sum of switching costs and bandwidth costs reaches a minimum, *i.e.* we should consider separate VPCs but with some sharing.

Separate, long VPCs make sense if there is adequately repeated updating (as in our automatic methods), and high volumes of traffic (*i.e.* high statistical multiplexing gain) on all VPCs. Sharing is, however, more attractive if reasonable link utilisation cannot be achieved by a single traffic flow and should therefore be used for VPCs with low volumes of traffic. So we leave end-to-end

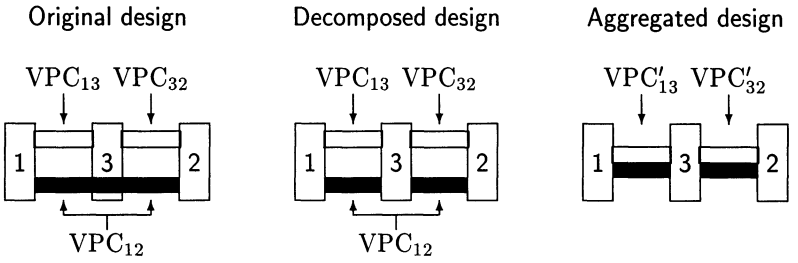

Figure 2 VPC network design modification by decomposition and aggregation.

VPCs with high volumes of traffic (for which efficiency is not a big problem) but allow end-to-end VPC switch low traffic volumes to be decomposed into at most two VPC segments which can be aggregated with other, identical segments to allow for a higher degree of sharing. The principle of decomposing and aggregation is shown in Figure 2. The figure shows a network with three nodes (named 1, 2, and 3) and two links (from 1 to 3 and from 3 to 2 respectively). The left picture left shows a network with three VPCs, the one in the middle shows how one VPC is decomposed into two segments, and right one how the two segments are aggregated with existing, identical VPCs.

2.2 Formal Notation and Terminology

We consider VPC networks of N nodes with known capacity matrices C (of size $N \times N$) where $c_{o,t}$, $o,t : (o,t = 1,\ldots,N,\ o \neq t)$ denotes the available bandwidth from node o to node t. All nodes contain VP cross connects and VC switching systems. This means that they may originate, terminate, and relay traffic either as bundles (VPs) or channels (VCs). To originate or terminate a channels requires both VP and VC functionality, while relaying can be done on the VP level only in the VP cross connect, or on the VC level by the VC switching system if preceded by VP demultiplexing and followed by VP multiplexing.

For the sake of simplicity we omit conditioning on service classes and limit ourselves to the case of a single, uniform service class. (Note that we deploy service class (or traffic type) separation between VPC networks. Since we study decomposition and aggregation of VPCs within such VPC networks, neither the number of service classes (or traffic types) studied nor the specific choices will impact our results from a qualitative point of view.)

All nodes originate traffic to and terminate traffic from all other nodes but themselves. User demands are fully characterised by a sequence of known end-to-end traffic demand matrices $A(k)$ (of size $N \times N$), where $a_{o,t}(k)$ denotes

the traffic demand from o to t at time k, $k : (k = 1, \ldots, K)$. The time index k indicates intervals such as hour, day of week, or day of year.

For each traffic matrix there is a corresponding VPC network design matrix $D(k)$ (of size $N \times N$), the elements $d_{o,t}(k)$ of which contain physical routes and associated bandwidths (in terms of circuits) for the end-to-end traffic demand $a_{o,t}(k)$. $D(k)$ is computed from $A(k)$ taking the relevant QoS demands for the service class(es) and the constraints of the physical network C, for details of the algorithm see (Arvidsson 1995). Our task is now to decompose the lightly loaded individual VPCs of all designs into two component VPCs of shorter length, which then must be aggregated to form shorter, common VPCs with higher loads.

To simplify matters, we judge VPC traffic volumes by a simple threshold value (expressed in number of circuits). VPCs above this value are called "full VPCs" and are left untouched; VPCs below the value are called "thin VPCs" and are candidates for decomposition and aggregation. Typically, the threshold is set such that VPCs which can benefit significantly from increased volumes are affected. The goal of our decomposition procedure is to refine a design $D(k)$ into a new, modified design $D'(k)$ which is free from thin VPCs.

3 RECURSIVE DECOMPOSITION AND AGGREGATION

Our proposed algorithm examines the thin VPCs one by one, and for each of them it tries various decompositions until both segments can be aggregated into full VPCs. Segments can be aggregated with existing, full VPCs or with existing thin VPCs or other segments if the total bandwidth after the aggregations makes the thin VPC/segments qualify as full VPCs. To allow for the latter option must the outcome of a particular attempt depend on the outcome of the following attempts. The algorithm therefore takes a recursive approach to decomposition and aggregation, where a particular attempt is judged only when all offsprings of that attempt can be judged.

We will now describe the details of algorithm. The basic components are four data lists, a main procedure, and a set of supporting functions. The lists are `fullVPClist`, which contains all full VPCs; `thinVPClist`, which contains all thin VPCs; `candVPClist`, which contains all segments and merged segments that have not yet qualified as a full VPC; and `failVPClist`, which contains all thin VPCs which cannot be modified. To speed up the decomposition and aggregation, the lists in our implementation are sorted lexicographically, both forwards and backwards.

The `main` procedure first initiates the lists mentioned above and then enters a loop where the thin VPCs are examined for decomposition and aggregation one by one. For each thin VPC, any node but the first and last ones may act as decomposition points. Nodes are tried sequentially for decomposition until one that results in successful aggregation has been found:

```
procedure main
  form fullVPClist and thinVPClist from design;
  clear candVPClist and failVPClist;
  let selectedPath be the first member of thinVPClist;
  let breakNode be the node after firstNode;
  repeat
    if successful(selectedPath,breakNode) then
      implement pending decompositions and aggregations;
      let selectedPath be the first member of thinVPClist;
      let breakNode be the node after firstNode;
    else if breakNode is not the node before lastNode then
      restore pending decompositions and aggregations;
      let breakNode be the next node;
    else
      restore pending decompositions and aggregations;
      move selectedPath from thinVPClist to failVPClist;
    endif
  until selectedPath is undefined
  stop;
endprocedure
```

The function `successful(path,node)` returns `true` or `false` depending on whether the two segments resulting from decomposing `path` at `node` will can be successfully aggregated or not:

```
function successful(selectedPath,breakNode)
  let prefixPath be selectedPath from firstNode to breakNode;
  let suffixPath be selectedPath from breakNode to lastNode;
  if match(suffixPath,breakNode) and match(prefixPath,breakNode) then
    return true;
  else
    return false;
endfunction
```

The function `match(path,node)` returns `true` or `false` depending on whether `path` can be aggregated or not. Successful aggregation can result either from a match to an existing member of `fullVPClist`, or from a match to a member of `candVPClist` such that aggregation results in that the member qualifies as a full VPC. The function first tries the former option, then the latter one, and returns with a negative result if both options fail:

```
function match(subpath,breakNode)
  if fullVPClist contains a path equal to subpath then
    set pending implementation;
    return true;
  else if extendible(subpath,breakNode) then
    return true;
```

```
  else
    return false;
  endif
endfunction
```

The function `extendible(path,node)` handles the segments which may become full VPCs in `candVPClist`. It returns `true` or `false` depending on whether aggregating `path` with other segments results in full VPCs or not. These other segments may already be in `candVPClist` or tracked down and inserted by scanning the remaining members of `thinVPClist`. The first step is to find `path` in `candVPClist`. If this is successful, `path` is added to the existing member, otherwise `path` is entered as a new member. In the former case may the existing member now qualify as a full VPC, in which case the aggregation is successful. Otherwise more segments must be identified from `thinVPClist` and added on to result in a successful aggregation. The procedure therefore scans the latter list for members where a segment equal to `path` may be formed. The scanning continues until a new, full VPC can be formed, or until all members of `thinVPClist` have been tried. Each member scanned is tested for successful aggregation of both segments in the same way as in the main procedure, *i.e.* we may apply recursion:

```
function extendible(subpath,breakNode)
  let mergePath be the member of candVPClist equal to subpath;
  if mergePath is defined then
    let mergePath be the aggregate of mergePath and subpath;
    if mergePath qualifies as a member of fullVPClist then
      set pending implementation;
      return true;
    endif
  else
    make subpath a new member of candVPClist;
  endif
  repeat
    let mergePath be the next member of thinVPClist
    with a segment equal to subpath;
    if mergePath is defined then
      if successful(mergePath,breakNode) then
        set pending implementation;
        return true;
      endif
    endif
  until mergePath is undefined
  reset pending implementation;
  return false;
endfunction
```

4 NUMERICAL RESULTS

4.1 Test Scenario

(a) Networks, Traffics, and Tools

To facilitate numerical tests, a computer programme was used to generate eight distinct networks of $N = 20$ nodes and $K = 8$ distinct traffic demand matrices for each network. Requests for connections arrive according to independent Poisson processes for each origin-termination (OT) pair. The connection holding time is assumed to be negative exponentially distributed with unit mean. User demands are uniformly distributed between about 5 and 350 Erlangs per OT pair, with a difference of about $\pm 20\%$ per OT pair from one matrix $A(k)$ to another $A(k')$ (corresponding to similar traffic demand variations over the time of the day, the day of the week *etc.*). Network transmission capacities C are set to allow for a VPC network configuration $D(0)$ with one VPC per OT pair (along the shortest physical route) with a capacity that allows a probability of rejection of exactly 10^{-2}, for a traffic which is the average over the whole range of traffic matrices $A(1), \ldots, A(K)$. To give an idea of the actual test networks, we provide an example of a network and a traffic matrix in appendix 1.

Next, a network simulator was constructed which implements any test network according to its capacity matrix C and an associated traffic matrix $A(k)$. To simulate traffic dynamics, user demands change every $T = 30$ time unit by replacing a traffic matrix $A(k)$ by its successor $A(k+1)$ in a cyclic fashion such that $A(K)$ is followed by $A(1)$.

(b) Congestion Control and Routing

Requests for a connection to a node d arriving at a node o are accepted if there is enough free bandwidth available, *i.e.* if the number of connections in progress is less than the allocated capacity (recall that bandwidths are expressed as circuits). As indicated before, there may be more than VPC for every OT pair and all options are tried for all requests. Paths which have been modified into two hops are tried last and require two VCs, one per hop. Connections over one hop paths are said to be **direct** while those over modified paths are said to be **broken**. Requests which cannot find free bandwidth on any of these options are **rejected**.

To reduce the probability of rejection we can allow rejected requests to hunt for free bandwidth on two VPCs in series, *i.e.* along a two-hop path where the first hop is from the origin to an arbitrary intermediate node and the second one from the intermediate node to the termination. Noting the apparent similarities to overflows in circuit switched networks, we have adopted the dynamic alternative routing method (DAR) of selecting the intermediate node (Gibbens *et al.* 1989) and deployed trunk reservation (Katschner 1974)

in order to prevent excessive, inefficient use of this facility. Connections over such paths are referred to as **overflowed**. To limit the number of intermediate nodes per connection are paths which have been modified not available for this feature.

A modified path also constitutes two direct paths: one between the origin and the intermediate node, and another one between the intermediate node and the termination. To achieve maximal statistical sharing, we give connection requests between these two pairs full access to the bandwidth provided by the modified path (*i.e.* it is not reserved to the original OT pair). This means that connections between the intermediate node and either end node can make a modified VPC unable to accept end-to-end connection requests, even if less than the engineered number of end-to-end connections are in progress. Request for end-to-end connection which fail in this way are said to be **blocked**.

(c) Virtual Path Management

In the off-line approach to VPC management considered here are the K traffic matrices and their times of occurrence assumed to be known in advance. This allows the K VPC network designs to be computed in advance and implemented in the network as traffic change, *i.e.* design $D(k)$ is followed by design $D(k+1)$ *etc.* and design $D(K)$ by design $D(1)$.

Changing VPC network designs involves connecting, modifying, and closing VPCs. A **VPC connection** is when a new physical route is opened between two nodes by inserting entries in the routing tables of the VPC switching entities at all nodes along the route, A **VPC modification** is when an existing physical route between two nodes is kept but the bandwidth allocated to it is changed by the replacing the contents in the CAC tables of the VC switching entities at the two end nodes, and a **VPC disconnection** is when an existing physical route between two nodes is closed by removing entries from the routing tables of the VP switching entities at all nodes on the route.

Changing VPC designs may lead to a situation of **bandwidth violation**. This means that the number of connections on a physical link exceeds its capacity, *i.e.* the number it can support at a given cell level QoS. Bandwidth violation thus means that cell level QoS is impaired, and may happen if a new VPC network design means more bandwidth for some VPCs and less for others. A shortage (*i.e.* a violation) will then occur if (i) the number of connections in progress on VPCs subject to a bandwidth decrease exceeds the new bandwidth and (ii) their new, lower limits are reached slower than the number of connection in progress on VPCs subject to a bandwidth increase move towards their new, higher limits. In general, the impact of a bandwidth violation depends on the degree of violation and time during which it persists. The problem can be addressed in many ways; *e.g.* by physical rerouting, where the excess connections are physically rerouted while in progress (as is done for hand overs in cellular, mobile systems) and stay there for their remain-

ing time; virtual rerouting, where excess connections are logically rerouted by requesting non-saturated VPCs over the same links to increment their occupancy during their remaining time of the excess connections; by policing, where police mechanisms will impose higher cell losses to make excessive streams conform with the imposed bandwidth requirements; or by ignorance, where actual violations (*i.e.* on links where both conditions above are fulfilled) will suffer from cell losses.

Each option is associated with costs, in the first two the costs refer to processing of all potentially dangerous calls, in the third one the costs refer to cell losses for all calls in the potentially violating flow, and in the fourth one the costs refer to cell losses for all flows on actually violated links. (Note that much fewer cells thus will be lost with option four than with option three!)

4.2 Algorithm Performance

(a) Static performance

Clearly, the outcome of the algorithm is dependent on the order in which thin VPCs are tried. We have adopted the approach to treat the thin VPCs in descending order of the path length (that is, the number of transmission links in a VPC). The rationale behind this is that in general it is harder to decompose a long VPC into two segments and hence it is better to deal with these at an early stage when more alternatives are available by way of recursion of other thin VPCs. In addition to processing the thin VPCs in descending path length, for the same path length we process the thin VPCs in ascending order of offered traffic or allocated circuits (as the case may be) since it is expected it is harder to accumulate traffic to satisfy the traffic threshold requirement for a VPC with a smaller volume of traffic.

We have tested our procedure on the VPC designs $D()$ for all 64 network and traffic configurations. The initial designs contain between 300 and 400 VPCs. Using a threshold value of 30 circuits, the numbers of thin VPCs lie in the range of 100 to 150.

After applying the procedure, the numbers of unmodified thin VPCs lie in the range of 0 to 16, the average number is about 6.5. The numbers of new full VPCs formed lie in the range of 10 to 31, with an average of about 19. This means that for these originally thin VPCs, no intermediate switching of their end-to-end traffic is required because of success in traffic aggregation from longer thin VPCs that use these VPCs as one of their two segments. The numbers of new VPC formed lie in the range of 0 to 9 with an average of about 3 (recall that new VPC are formed to carry transit traffic only).

For two test cases, we have inverted the order of processing thin VPCs, that is, we process the thin VPCs in ascending order of path length. In one test case, the number of unmodified thin VPCs has gone up from 6 to 10. In another case, the number of unmodified thin VPCs has gone up from 7 to 12.

Table 1 Total results expressed as occurrences.

Metric used	Direct		Overflow	
	Mod.	Ori.	Mod.	Ori.
VCC rejected (%)	1.4692	1.6896	0.7316	0.8679
VCC broken (%)	1.6445	—	1.6890	—
VCC blocked (%)	0.0451	—	0.0796	—
VCC overflowed (%)	—	—	0.9901	1.2281
VPC modifications (%)	0.0239	0.0279	0.0239	0.0279
VPC connections (%)	0.0187	0.0391	0.0187	0.0391
VPC disconnections (%)	0.0186	0.0389	0.0186	0.0389
Bandwidth violation (%)	0.0009	0.0008	0.0020	0.0017

(b) Dynamic Performance

Simulating each network for 4,800 time units (corresponding to 20 cycles of traffic patterns or about 150 million connections), we obtain the results shown in Table 1. The large number of connections per network means that the confidence obtained for each network is very high. To limit the amount of data, the tables show averages over all networks. The variance that follows from network differences is omitted since we do not think it is particularly significant to this work but would make the tables harder to read.

The columns refer to specific combinations of strategy for routing (direct routing only or overflow routing applied) and design (modified or original). The rows are divided into three groups, the first one relate to the handling of VCCs, the second one to the handling of VPCs, and the third one to bandwidth violations.

The first group gives the fractions of requests resulting in rejected, broken, blocked, and overflowed connections respectively. As expected, it is seen that both modification and overflow routing reduces rejection and consequently improves utilisation. With modified networks, a little over 1.5% of all requests try broken connections, and that the number increases if overflow routing is used. The small number follows from the restrictive usage in terms of modification threshold and path search order, and the increase is a result of more extensive usage following from the overflow option. Blocking events exhibit a similar behaviour, where approximately one request out of 2,000 arriving ones (or one out of 30 for which modification is tried) is blocked. With overflow routing, about 1% of all requests are handled as overflowed connections. The small number follows from the ability of the networks to satisfy most requests as direct connections, and the lower value noted for modified designs follows from the fact that overflow routing has less to add when design modification already has made more circuits available to a wide range of OT pairs.

Table 2 Revenues and expenses assumed in the performance evaluation.

Action	Gain	Cost
Carrying a direct VCC	1.00	
Carrying a broken or overflowed VCC	1.00	0.01
Performing a VPC connection (per node)		0.10
Performing a VPC modification (per node)		0.10
Performing a VPC disconnection (per node)		0.10
Bandwidth violation (per link, percentage, and time)		10,000.00

The second group contains the number of modifications, connections, and disconnections of VPCs per generated call request, and the average degree of bandwidth violation. It is seen that the number of modifications, connections, and disconnections are independent of the VCC routing strategy but drop somewhat with modification. The first observation follows immediately from the fact that the VPC network design algorithm does not take the routing strategy into account. The second observation is related to the decrease in number of VPCs in modified designs, a conclusion which is supported by noting that the drop is strong for connections and disconnections, but only weak for modifications.

Finally, it is seen in the third group that bandwidth violations are marginal in all cases. The larger numbers noted for modified designs and overflow routing respectively and in combination follow from the associated higher utilisation.

Our overall purpose is to maximise network profit, *i.e.* the difference between revenues and expenses. Revenues come from charging customers for the usage of services, and expenses are associated with maintaining the network and providing the services. To obtain an overall metric of network profit we introduce a monetary unit which is used to express all revenues and expenses. Our choices, inspired by discussions with operators, are summed up in Table 2. Although one can think of different values, we feel that the order of magnitude of our choices is reasonable, and they allow us to capture all aspects of our study into a single metric.

The monetary unit is set such that a direct connection represents a gain of 1.00 at no cost. A broken or overflowed connection represents the same gain, but with an additional cost of 0.01 to account for the additional overhead and reduced QoS associated with a second VC. VPC management actions represent costs of 0.10 per node (corresponds to ten times the cost of a VC action, a relatively high number we believe). Finally, bandwidth violation represent a cost of 10,000.00 per link, per relative degree of violation, and per

Table 3 Total results expressed as monetary units.

Metric used	Direct		Overflow	
	Mod.	Ori.	Mod.	Ori.
Revenues per time unit	30140.61	30139.85	30140.48	30139.54
Expenses per time unit	458.88	516.50	248.50	280.75
Profitability (%)	98.48	98.29	99.18	99.07
Improvement potential (%)	1.52	1.71	0.82	0.93

Table 4 Separated results expressed as monetary units.

OT pair category	Metric used	Direct		Overflow	
		Mod.	Ori.	Mod.	Ori.
Direct paths	Profitability (%)	98.84	98.55	99.30	99.17
	Improvement potential (%)	1.16	1.45	0.70	0.83
Mixed paths	Profitability (%)	97.92	97.88	98.91	98.86
	Improvement potential (%)	2.08	2.12	1.09	1.14

time unit. Table 3 shows the same results as in Table 1 expressed as monetary units.

The first group of rows presents the absolute results per time unit and the second group gives some normalised results. In the first group it is seen that the networks are offered about 30,000 Erlangs, and the expenses per time unit are small compared to the revenues. The second group considers the actual profit (revenues minus expenses) in relation to the theoretical optimum (maximal revenues and no expenses). It is seen that all four cases are close to full profitability, but what is more important is the remaining improvement potential. Comparing to the case of neither modification nor advanced routing with an improvement potential of about 1.7%, deploying both of them exploits about half that potential down to 0.82%. It is also seen that the improvement of advanced routing (0.78%) is larger than that of modification (0.19%), and that the two features are partly overlapping since the total improvement of both actions (0.89%) is less than the sum of the individual improvements (0.97%).

To get a deeper understanding of the results, we have also conducted separate measurements for OT pairs with and without modified VPCs in their designs $d_{o,t}(k)$. Table 4 gives the results in monetary units for OT pairs with direct VPCs only (upper rows) and with some modified VPCs (lower rows).

The most important conclusion is that both classes of OT pairs benefit from

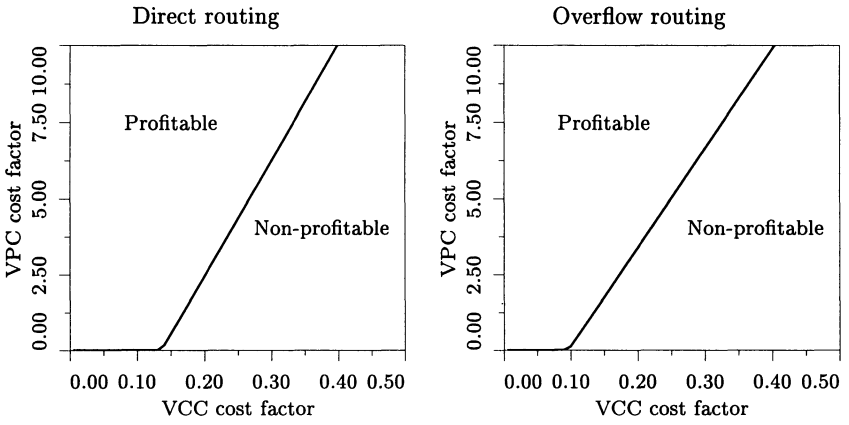

Figure 3 Profitability of modification *vs.* VCC and VPC costs for direct routing (left) and overflow routing (right).

modification and advanced routing. It is also interesting to note that OT pairs without modified VPCs benefit the most from modification, an observation which is attributed to the fact that more bandwidth is made available to them, while OT pairs with modified VPCs, which require free bandwidth on two VPCs at the same time, cannot access the bandwidth to the same extent. On the other hand the situation is reversed with advanced routing, an observation which leads to the conclusion that generally disadvantaged OT pairs are more likely to have their paths modified.

Finally, we study the sensitivity of our conclusions with respect to costs. The diagrams in Figure 3 show the cost ranges in which modification is profitable for simple and advanced routing respectively. VCC costs refer to the additional expense associated with broken or overflowed VCCs, while VPC costs refer to the expense associated with connecting, modifying, or disconnecting a VPC. As expected, it appears that modification is always profitable if the VCC cost is little or none, and that the higher the VCC cost, the higher must the VPC cost be for modification to make sense.

5 CONCLUSIONS AND FURTHER WORK

This work represents a first attempt at investigating the trade off between VPC and VCC switching. Earlier works in the area, *e.g.* (Burgin 1989), have come to the same conclusion, *i.e.* that VPC and VCC switching should be used in combination, but this study is, to the author's knowledge, the first with this level of detail and the first to study dynamic aspects. Building on our earlier results on optimal management on fully interconnected VP networks, we have proposed a method to identify and eliminate inefficient VPCs. We

have also been able to demonstrate how the method can be applied successfully to improve network profitability.

An obvious issue for further investigations is the question of optimal modification in terms of chosing the best threshold. Looking at the *pros et cons* of VPC and VCC routing, it is obvious that this threshold must depend on the relationship between VPC and VCC management costs as well as on the multiplexing characteristics of each service class.

Another interesting issue is to do iterative designs in which the original traffic demand matrix and the new network design matrix are combined into a new traffic demand matrix which again is subject to VPC network design followed by possible modification. The same procedure may then be repeated over and over again until the demand matrices converge.

A further point of interest is to apply a threshold to a complete OT pair rather than a single VPC. In this way are inefficient flows handled in a more collected effort. Finally, we would like to conduct a series of test for the case where VPC designs are computed in real time based on on-line estimations of traffic demands.

Finally, the complexity of our algorithm, and alternative algorithms are strong points of interest.

REFERENCES

Anick, D., Mithra, D., and Sondhi, M. (1982) Stochastic Theory of a Data-Handling System with Multiple Sources, *Bell System Technical Journal*, **61**, 1871–94.

Arvidsson, Å. (1994) Real Time Management of Virtual Paths, in *Proc. IEEE Globecom 1994*, **3**, 1399–1403, IEEE.

Arvidsson, Å. (1995) High Level B-ISDN/ATM Traffic Management in Real Time, in *Performance Modelling and Evaluation of ATM Networks*, **1**, 177–207, Chapman & Hall.

Burgin, J. (1989) Management of Capacity and Control in Broadband ISDN, *Int. J. of digital and Analog Cabled Systems*, **2**, 155–65.

Gibbens, R., Kelly, F., and Key, P. (1989) Dynamic Alternative Routing — Modelling and Behaviour, in *Proc. 12th Int. Teletraffic Cong.*, Turin (Italy), paper no. 3.4A.3.

Katschner, L. (1974) Service Protection for Direct Final Traffic in DDD-networks, *Nachrichtentechnische Zeitschrift (NTZ)*, **5**, 480–484.

ACKNOWLEDGEMENT

The author wishes to acknowledge that the algorithm presented is devised and implemented by Dr. Yiu Kwok Tham during his stay with the Department of Telecommunications and Mathematics at the University of Karls-

krona/Ronneby. The first report on this work was prepared jointly by him and the current author. However, despite repeated attempts, the present author has been unable to get in touch with Dr. Tham for the preparation of the current version.

APPENDIX 1 TEST NETWORKS

Below we give an idea of the kind of test networks used by showing a sample topology, Figure 4, a sample capacity matrix, Table 5, and a sample demand matrix, Table 6. A complete description of the algorithm used to generate the networks is given in (Arvidsson 1995).

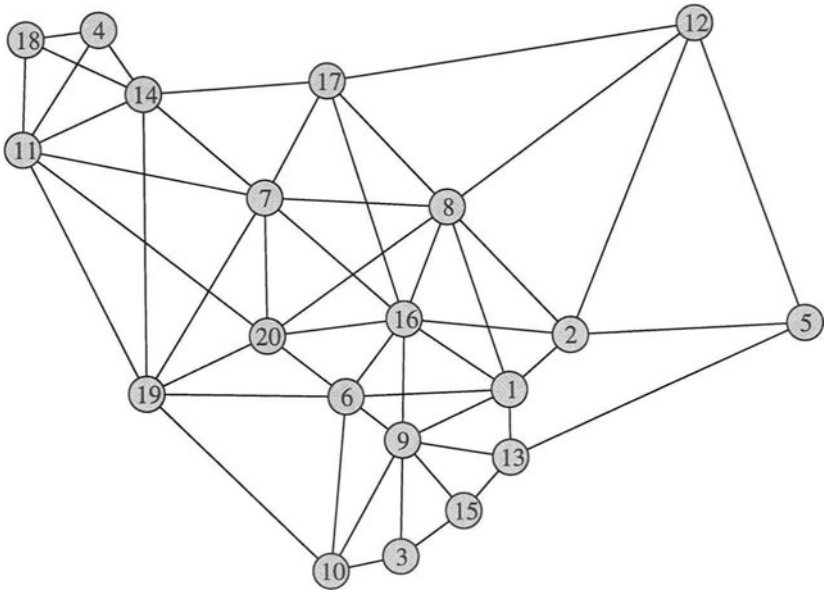

Figure 4 A sample topology.

Table 5 A sample capacity matrix.

OT	C	OT	C	OT	C	OT	C	OT	C
1– 2	2570	1– 6	600	1– 8	670	1– 9	1560	1–13	2160
1–16	1510	2– 5	1320	2– 8	500	2–12	1190	2–16	2270
3– 9	1450	3–10	1690	3–15	510	4–11	540	4–14	2560
4–18	90	5–12	1350	5–13	1090	6– 9	2760	6–10	330
6–16	410	6–19	740	6–20	3130	7– 8	1540	7–11	1350
7–14	4030	7–16	4540	7–17	500	7–19	610	7–20	650
8–12	930	8–16	580	8–17	810	8–20	560	9–10	1070
9–13	1410	9–15	2280	9–16	2980	10–19	2070	11–14	590
11–18	1140	11–19	960	11–20	2030	12–17	2170	13–15	1610
14–17	2360	14–18	2130	14–19	1580	16–17	1200	16–20	1060
19–20	800								

Table 6 A sample demand matrix.

OT	A	OT	A	OT	A	OT	A	OT	A
1– 2	156.6	1– 3	131.6	1– 4	190.2	1– 5	85.6	1– 5	85.6
1– 6	46.2	1– 7	44.9	1– 8	30.6	1– 9	54.4	1–10	140.9
1–11	111.5	1–12	223.1	1–13	113.8	1–14	42.4	1–15	72.6
1–16	35.4	1–17	77.8	1–18	53.1	1–19	132.5	1–20	124.4
2– 3	313.4	2– 4	135.6	2– 5	18.4	2– 6	90.1	2– 7	11.5
2– 8	93.6	2– 9	71.0	2–10	37.3	2–11	162.6	2–12	136.7
2–13	204.2	2–14	27.4	2–15	202.7	2–16	178.4	2–17	257.2
2–18	36.0	2–19	305.4	2–20	207.9	3– 4	283.2	3– 5	121.1
3– 6	35.0	3– 7	20.8	3– 8	23.3	3– 9	64.2	3–10	315.9
3–11	270.3	3–12	206.5	3–13	163.1	3–14	276.6	3–15	80.5
3–16	191.6	3–17	106.4	3–18	234.3	3–19	195.9	3–20	339.5
4– 5	52.5	4– 6	282.1	4– 7	257.6	4– 8	148.4	4– 9	160.9
4–10	13.6	4–11	142.7	4–12	273.6	4–13	134.1	4–14	33.2
4–15	11.6	4–16	77.2	4–17	22.4	4–18	67.5	4–19	256.9
4–20	40.0	5– 6	112.1	5– 7	151.2	5– 8	24.4	5– 9	311.8
5–10	14.4	5–11	205.0	5–12	261.8	5–13	9.4	5–14	245.5
5–15	264.6	5–16	204.4	5–17	161.8	5–18	99.3	5–19	48.5
5–20	118.5	6– 7	248.3	6– 8	50.6	6– 9	249.3	6–10	272.3
6–11	118.4	6–12	28.2	6–13	104.7	6–14	244.3	6–15	252.4
6–16	167.0	6–17	126.4	6–18	246.0	6–19	15.9	6–20	120.1
7– 8	128.0	7– 9	304.4	7–10	108.2	7–11	85.8	7–12	246.5
7–13	32.4	7–14	150.1	7–15	247.1	7–16	274.7	7–17	70.5
7–18	191.6	7–19	195.2	7–20	44.2	8– 9	60.9	8–10	60.4
8–11	175.7	8–12	191.6	8–13	242.3	8–14	294.7	8–15	106.8
8–16	245.9	8–17	277.8	8–18	250.6	8–19	22.0	8–20	226.6
9–10	151.2	9–11	259.1	9–12	274.9	9–13	161.3	9–14	13.6
9–15	200.7	9–16	328.8	9–17	175.3	9–18	53.7	9–19	108.4
9–20	322.5	10–11	155.9	10–12	199.5	10–13	230.6	10–14	297.6
10–15	100.3	10–16	13.4	10–17	151.4	10–18	105.1	10–19	39.8
10–20	111.5	11–12	171.1	11–13	149.3	11–14	210.2	11–15	49.0
11–16	265.2	11–17	102.8	11–18	26.6	11–19	50.5	11–20	334.9
12–13	92.2	12–14	251.9	12–15	98.0	12–16	10.4	12–17	141.4
12–18	213.1	12–19	198.4	12–20	72.3	13–14	11.0	13–15	281.2
13–16	174.5	13–17	27.1	13–18	234.8	13–19	71.6	13–20	223.2
14–15	156.7	14–16	115.6	14–17	203.6	14–18	157.6	14–19	132.8
14–20	154.5	15–16	209.5	15–17	276.4	15–18	45.3	15–19	133.2
15–20	328.1	16–17	159.2	16–18	192.1	16–19	169.2	16–20	132.2
17–18	138.2	17–19	239.7	17–20	119.9	18–19	73.4	18–20	298.1
19–20	6.4								

BIOGRAPHY

Åke Arvidsson received his Ph.D. from the Lund Institute of Technology in Lund, Sweden, in 1990. He is currently acting professor at the Department of Telecommunications and Mathematics, University of Karlskrona/Ronneby, Sweden. Current research interests include bandwidth management and traffic routing in ATM networks, traffic modelling for buffer engineering and call acceptance control in ATM networks, and congestion control mechanisms for intelligent networks. His URL is http://www.itm.hk-r.se/~akear.