Remove $\psi'$ from $\Psi$.
Mark $\psi'$ as delete-pending.
Recalculate the optimal VC layout on $\Psi$.
until $\text{Loss}(\psi') > \text{Loss}_{\text{max}}$

Algorithm DA (Distributed VP Addition):

0 Upon receipt of VC setup/teardown requests do:
1 Maintain $EU(p)$ for every potential VP $p \in \mathcal{P}(G_{\Psi})$ for which
2 (1) node $v$ is an endpoint (i.e. $v \in \varepsilon(p)$) and
3 (2) $p \subseteq \lambda$ for some VC $\lambda \in \Lambda$
4 Upon receipt of the ROC-PDU at node $v$ do:
5 Add the VPs from ROC[v].NewVPs to the local view on the VP layout
6 Remove own VPs from ROC-PDU (i.e. ROC[v].NewVPs $\rightarrow \emptyset$)
7 loop forever
8 Find a potential VP $\psi'$, for which $\text{Gain}(\psi')$ is maximal
9 If $\text{Gain}(\psi') < \max \text{ROC[v].MaxGain}$ then
10 Update ROC[v].MaxGain $\rightarrow \text{Gain}(\psi')$
11 Send ROC-PDU to the next node in the ring
12 Exit the procedure.
13 Add $\psi'$ to the local picture of $\Psi$
14 Add $\psi'$ to ROC-PDU (i.e. ROC[v].NewVPs $\rightarrow$ ROC[v].NewVPs $\cup \{\psi'\}$
15 Start a VP setup procedure for $\psi'$
16 end loop
The notion of VC/VP switch is formalized by the following definition:

**Definition 10 (VP/VC switch).** A VC switch \( v \) with respect to a VC \( \lambda \), is an intermediate node, which is an endpoint to a VP which is used by the VC, i.e. \( \psi \in \mathcal{R}(\lambda, \Psi) \), \( v \in \varepsilon(\psi) \). A VP switch with respect to \( \lambda \) is a node along the physical route that \( \lambda \) takes, which is not a VC switch.

The next definition formalizes the optimal VP layout design problem, given the traffic pattern as captured by the VC layout.

**Definition 11 (Optimal VP layout Design).** Given a network \( G \), a VC layout \( \Lambda \), a VC routing \( \mathcal{R}(\Lambda, G) \), and an integer \( h > 0 \), find a VP layout \( \Psi \) with minimal size \( |\Psi| \), for which \( \mathcal{H}(\Lambda, \Psi) < h \) (for the average case, alternatively \( H(\Lambda, \Psi) < h \) for the worst case).

The problem of minimizing \( \mathcal{H} \), seems harder than that of minimizing the worst-case hop count \( H \), since it is more sensitive to local changes, that do not effect the worst-case. As the problem for the worst-case is NP-complete \([7]\), we conjecture the problem for the average-case to be NP-complete as well.

We now present formal versions of the CA, CD, and DA algorithms.

**Algorithm CA (Centralized VP Addition):**

0 loop
1 For every sequence of VPs \( s = (\psi_1, ..., \psi_k) \) with the following properties:
2   (a) \( s \) is part of some VC (i.e. \( \exists \lambda \in \Lambda : s \subseteq \mathcal{R}(\lambda, \Psi) \)),
3   (b) \( s \) is not equal to any existing VP
4     (i.e. \( \forall \psi \in \Psi : \mathcal{R}(\psi_1, G) \cdot \cdots \cdot \mathcal{R}(\psi_k, G) \neq \mathcal{R}(\psi, G) \)).
5 do:
6       Compute the expected utilization of \( s \), by \( EU(s) = |\{\lambda \in \Lambda | s \subseteq \mathcal{R}(\lambda, \Psi)\}| \)
7       Compute the gain of \( s \) (in terms of \( \mathcal{H} \)), by \( Gain(s) = (|s| - 1) \cdot EU(s) \)
8     Find a sequence \( s' \), for which \( Gain(s') \) is maximal
9     Add a new VP \( \psi \) to \( \Psi \) with the same physical route as that of \( s' \)
10    Mark \( \psi \) as add-pending.
11    Update the route of VCs that used \( s' \) (i.e. \( s' \subseteq \mathcal{R}(\lambda, \Psi) \)),
12      to include the new VP instead of \( s' \).
13 until \( Gain(s') < Gain_{\text{min}} \)

**Algorithm CD (Centralized Deletion):**

0 loop
1 For every existing VP \( \psi \in \Psi \setminus \Psi_{\text{Static}} \) do
2     Compute the utilization of \( \psi \), by \( U(\psi) = |\{\lambda \in \Lambda | \psi \in \mathcal{R}(\lambda, \Psi)\}| \)
3     Let NOPA(\( \psi \)) denote the Next Optimal Path After \( \psi \)
4     (namely the shortest path in \( \Psi \setminus \{\psi\} \)
5     with the same underlying physical route as that of \( \psi \))
6     Compute the loss of \( \psi \) by \( Loss(\psi) = (|\text{NOPA}(\psi)| - 1) \cdot U(\psi) \)
7     Find a VP \( \psi' \), for which \( Loss(\psi') \) is minimal
In this appendix we give formal definitions which correspond to those in Section 2. We also present formal versions of the algorithms, which were informally presented in Section 3. Given an ATM network, we model it by an undirected graph $G = (V, E)$ where $V$ is the set of nodes and $E$ the set of physical links between them.

**Notation.** We use the following notations:

1. Let $\mathcal{P}(G)$ be the set of all simple paths in $G$.
2. For a path $p \in \mathcal{P}(G)$, let $|p|$ denote the number of edges from which $p$ is composed, let $\varepsilon(p)$ be the set of two vertices which are the endpoints of $p$.
3. For two paths $a, b$ such that $\varepsilon(a) \cap \varepsilon(b) = \{x\}$, denote their concatenation (at $x$) by $a \bullet b$.

**Definition 6 (VP layout).** A *virtual path layout* is a collection of VPs $\Psi = \{\psi_1, ..., \psi_k\}$.

The function $\Re(\psi, G)$ maps a VP $\psi \in \Psi$ to a simple path in $G$ which corresponds to the physical route in the network along which $\psi$ is defined.

**Definition 7 (VC layout).** A *virtual channel layout* is a collection of VCs $\Lambda = \{\lambda_1, ..., \lambda_m\}$.

The function $\Re$ is extended to map VCs onto physical paths in $G$ as well, i.e. $\Re(\lambda, G)$ is a simple path in $G$ along which the VC $\lambda$ is defined.

Given a VP layout $\Psi$ and a VC layout $\Lambda$, each VC is routed using a sequence of VPs, the concatenation of which form the same physical path as that of the VC (it is assumed that such a mapping exists, since the VPL must support any VC layout). Again, we use the function $\Re$ to denote the mapping of a VC to a sequence of VPs, as follows:

**Definition 8 (routing).** Given a VC $\lambda \in \Lambda$ and a VPL $\Psi$, let the route that $\lambda$ takes with respect to $\Psi$ be $\Re(\lambda, \Psi) = (\psi_1, ..., \psi_h)$ such that:

1. $\psi_i \in \Psi$ for every $i = 1, ..., h$ ,
2. $\Re(\lambda, G) = \Re(\psi_1, G) \bullet \Re(\psi_2, G) \bullet \cdots \Re(\psi_h, G)$ and
3. $h$ is the minimal integer for which such a sequence exists.

**Definition 9 (hop count).** Given a VC $\lambda \in \Lambda$ and a VPL $\Psi$, the *VP hops* $h$ of $\lambda$ with respect to the VPL satisfies

$$h(\lambda, \Psi) = |\Re(\lambda, \Psi)|$$

The worst-case VP hops $H(\Lambda, \Psi)$ is defined by

$$H(\Lambda, \Psi) = \max_{\lambda \in \Lambda} h(\lambda, \Psi)$$

The average VP hops $\bar{H}(\Lambda, \Psi)$ is defined by

$$\bar{H} = \sum_{\lambda \in \Lambda} h(\lambda, \Psi) ; \quad \bar{H}(\Lambda, \Psi) = \frac{\bar{H}}{|\Lambda|}$$


starts at \( b \), advances on the old route, and clears the VC entries in the relevant VC tables (in both directions), at every intermediate VC switch in the old route.

**Observation 3.** If the VC switches of the old route are disjoint from those of the new route, then during the above protocol, the number of entries in the VC routing tables at every VC switch, does not exceed the number of VCs that use these routes. If the VC switches are not distinct, it is possible to split the routes into segments, in such a way that at every segment, the above property holds.

### 5 Bandwidth allocation

One important aspect which was not yet discussed in this paper is the necessary change in the allocation of bandwidth to VPs in the new VP layout. Assuming that initially there was a good allocation of bandwidth to the VPs, it only remains to redistribute this bandwidth among the VPs of the new layout.

Upon addition of a new VP, some of the bandwidth currently allocated to old VPs along its path, must be reallocated to it. The most reasonable reallocation policy is to calculate the ratio between the bandwidth of VCs that have used the old VPs, and the bandwidth of VCs that will use the new VP from the time it is added to the network. This ratio is used for reallocation of bandwidth. Similarly, upon VP deletion, its allocated bandwidth is reallocated to VPs along its path.

As for bandwidth management of VCs that are rerouted using the above IPRR protocol, during the IPRR setup phases, it is easy to calculate the total amount of occupied bandwidth of all the VCs that are rerouted together, and update the new VP switches with this information.

### 6 Summary

In this paper we have presented the problem of dynamically adjusting a layout of virtual paths, to the changing needs of the network users, as reflected by changes in the virtual channel layout. We proposed a centralized approach to the problem and a distributed approach and discussed their pros and cons.

We also showed a protocol for adjusting the current routing of VCs to the new layout without damaging the flow of data in the network, and discussed bandwidth allocation issues.

In large ATM networks that will emerge in the future, an automatic adjustment scheme of the network to the needs of its users — such as discussed in this paper — will significantly reduce the overhead of manual network management, and improve its response to changes and thus its performance.

### References


the initiating node $a$ transfers the list of VCsIs (termed VCI-List), of all VCs that use the first VP from the old sequence, to the other end of that VP. At the next node, the VCsIs are translated according to the VC routing tables, and only a sublist of VCI-List, that includes the VCs that use the second VP in the old route, is transferred along that VP. When the message reaches $b$, it contains the exact set of VCs to be rerouted. Since it is important to avoid mixing between these "parallel" VCs, VCI-List contains pairs of VCsIs, which consist of the original VCI of node $a$, and the transformed VCI (as described above). Therefore, at node $b$, the list contains pairs of VCsIs: a VCI at node $a$ and the VCI of the same VC at node $b$, which mark the correct association between parallel connections. We term these pairs at $b$, the association IDs of the VCs.

2. **Setup phase 1**: Node $b$ starts a setup phase on the new route, in which a set of VCsIs are transferred from a VC switch of the new route, to the next one (towards $a$), and are registered in the relevant VC routing table (very similar to a single VC setup protocol [5]). Along with these VCsIs, the message contains the association IDs for the VCs. When this message arrives at $a$, the new route of each VC in the direction of $b$ is ready, and $a$ redirects the VCs to that route, by changing the VPI and VCI fields in the relevant VC tables from the old route to the new route (the location in the table remains as before).

3. **Setup phase 2**: Node $a$ starts the same process in the opposite direction (again, as in [5]), but with the addition of the association IDs. When this message arrives at $b$, the reverse redirection process takes place (again, in accordance to the association IDs).

4. **VC takedown phase**: At this point, both sides have redirected the VCs to the new route, and the VC entries in the VP switches of the old route may be cleared. This is done by a phase that
2. This database is also updated with the new delete pending VPs so that they are not used by future VCs.

3. For each delete-pending VP that terminates at the current node, a counted is maintained, that counts VCs that still use the VP. When the counter reaches zero — the VP is deleted using a VP deletion protocol.

4. If such a counter does not reach zero within a timeout, the remaining VCs are rerouted to the new VP layout using the IPRR protocol.

4 In-path rerouting

In this section we present the "in path rerouting" protocol (IPRR for short). This protocol is used by the network adjustment algorithms (i.e. CNA or DNA) to reroute long-term VCs to use the new VP layout. The protocol takes as input two alternative VP sequences between a pair of network nodes, and reroutes VCs from the "old" sequence to the "new" one. This procedure differs from other rerouting protocols (e.g. [4, 5]), in that both old and new sequences use the same physical route in the network (but on different VPs). For this reason, it is possible to perform the rerouting of VCs in a way that is transparent to layers above ATM, with no additional mechanisms to preserve the low loss probability and the relative (FIFO) order of the cells (such as special buffers at AAL, and in-channel signaling mechanisms — as in [4]). This is possible due to the following observation.

Observation 2. The end-to-end FIFO order between cells that use two sequences of VPs on the same physical route is preserved, since a cell that advances using one sequence of VPs, uses the same switching elements, queues, and links as a cell that uses the other sequence.

IPRR is used for two distinct cases:

1. Adjustment of the network to the deletion of an existing VP: In this case, the old "sequence" is a single VP. This rerouting is necessary in the case that permanent VCs (or VCs which exist for long periods) use the delete-pending VP. Without IPRR, these VPs could never be deleted, which results in a VP layout which is not well adjusted to the needs of VC layout.

2. Adjustment of the network to the addition of a new VP: In this case, the new sequence is a single VP. The rerouting in this case releases other VPs from permanent VCs that use them, and enables the deletion of such VPs (which may become delete-pending in the future).

Despite some simplifications that are possible for each of these special cases, we present here a general case technique that unifies them, in which neither sequences is a single VP.

IPRR is a 4 way handshake protocol between the initiator - a (which is an end-node to both the old and new sequence), to the other end of the old and new sequences — b. The protocol proceeds as follows (see Figure 5):

1. **Associate VCI**s: This message has two functions: (1) To find out the list of VCs to be rerouted, (2) To associate the VCI in a and in b, that represent each VC so that the "parallel" VCs that go through a and b are not intermixed (see the association IDs below). To perform function (1),
Theorem 1. The DA algorithm has the following properties:

a. Correctness:

1. During the execution of the algorithm, whenever a node updates its local VP layout picture (at line 3) it has indeed the globally updated VP layout.

2. When a node selects a new VP to be added (at line 5), then this gain is maximal among all the nodes in the network, assuming the VC layout is stationary.

b. Emulation of CA: If the VC layout is stationary, the DA algorithm selects a sequence of pending VPs in the same order as may be produced in an execution of the CA algorithm.

c. Liveness: When the DA algorithm stops adding VPs to the VP layout, the gain of every potential VP is less than \( \text{Gain}_{\text{min}} \).

Proof. We prove each part of the theorem separately.

a. Correctness: To prove claim 1, recall that \( \text{ROC}^{[*]}.\text{NewVPs} \) contains all the VPs that were added in the last round (since every node \( v \) adds a VP to \( \text{ROC}^{[*]}.\text{NewVPs} \) whenever a VP is added to the network at line 3, and deletes its new VPs that were already seen by all other nodes at line 4).

To prove claim 2, note that a VP is added only if its gain is maximal among all the gains in \( \text{ROC}^{[*]}.\text{MaxGain} \). According to Lemma 1, the actual gain of these potential VPs has not increased since it was written in \( \text{ROC}^{[*]}.\text{MaxGain} \), thus the added VP is indeed maximal in its gain, assuming that the VC layout has not changed. □

b. Emulation of CA: By claim a(1), the local VP layout picture is the same in both the CA and DA algorithms, before selecting the next VP to be added. By the CA algorithm, it selects a new VP with maximal gain, by claim a(2), the DA algorithm selects a maximal VP as well. Thus there exists a way to select VPs by the CA algorithm which is identical to the choices made by the DA algorithm. □

c. Liveness: The DA algorithm does not stop if some node may add a VP with gain above \( \text{Gain}_{\text{min}} \).

Since (by claim a(1)) each node knows that exact VP layout, the proof follows. □

In its distributed version, the network adjustment algorithm is substantially simplified, since there is no need here for coordinated steps of VP setup/takedown and distribution of the new topology to the rest of the network. Instead, the algorithm uses ROC-PDU to distribute the relevant data and coordinate various asynchronous procedures at the network nodes.

Algorithm DNA (Distributed Network Adjustment):

0. The algorithm is activated in parallel to the distributed addition/deletion described above. The ROC-PDU is augmented by a flag per each new VP (in \( \text{ROC}^{[*]}.\text{NewVPs} \)) that is "true" if the VP setup protocol has completed for this VP.

1. Upon arrival of ROC-PDU at the node, new VPs for which the setup protocol has completed are added to the topology database used by the network layer for routing new VCs.
Note that the algorithm is invoked each time that the network layer receives a VC setup/takedown request. This happens at every node which is a VC switch for the new/deleted VC, while nodes that serve as VP switches to that VC are not aware of the setup/takedown event. This fact does not raise any difficulties, since the local picture at such nodes need not be affected by the event, as they are concerned only with adding VPs for which they are endpoints, based on the utilization by VCs for which they are VC switches.

The CA algorithm must keep (at line 1) for every potential VP, the number of VCs that will use it, in case it is added as a VP (i.e. its potential utilization). To maintain this information efficiently, each node must keep a counter per each route that goes through it, that holds the number of VCs which use that route, and are VC switched at it. We refer to this data as the VC layout local picture (VCLLP) at the node, and it can be maintained using a low overhead algorithm based on the following data structure (depicted in Figure 4). In this tree structure, each VC that is VC switched at a vertex and whose VC switches from its origin to its destination through the tree is v_1, v_2, ..., v_k adds one to the counter of a vertex v_1, whose route to v in the tree is v_1, v_2, ..., v_k.

The statistics according to which the gain/loss are computed at a vertex, is based on its VCLLP. Upon a setup of a new VC, that is VC switched at v, a setup request arrives at the network layer module of v. This request typically contains the VC switches along the path that will be used by the VC, and thus enables the network layer to find the appropriate route in its VCLLP, and update the utilization of the path accordingly. At VC takedown, a similar request arrives at v, which triggers the decrement of the relevant counter in VCLLP by one.

---

Note that a network node v may appear several times in the VCLLP of a node u (represented by several vertices), according to different routes which share both v and u. This imposes no special problems in the maintenance of the structure.
the Right Of Choice PDU (ROC-PDU for short), as only the node that currently holds it may select the next VP to be added. ROC-PDU contains an array with an entry for each node \( v \). Node \( v \) writes the gain of its best add pending VP in the entry (in field ROC[\( v \)].MaxGain), and thus the globally best VP to be added can be selected. The token also carries data on the VPs that were added during its previous round, to update the topological data at each node efficiently (in ROC[\( v \)].NewVPs). The structure of ROC-PDU is depicted in Figure 3.

The algorithm for VP addition at each node is as follows:

**Algorithm DA (Distributed VP Addition):**

1. Upon receipt of a VC setup/takedown request at the network layer of the current node - update the utilization of every potential VP for which the current node is an endpoint, and which may be used by the VC which is currently being set up/torn down (see definition of VCLLP in the sequel).

2. Upon receipt of the ROC-PDU at the current node \( v \) do perform steps 3 to 11 below.

3. Add the VPs from ROC[\( v \)].NewVPs to the local view of the current node on VP layout.

4. Remove the VPs of the current node from ROC[\( v \)].NewVPs (if any such VPs were added at the previous round).

5. Find a potential VP, for which the gain is maximal, amongst all the potential VPs whose endpoint is the current node.

6. If this gain is less than the maximum gain of the other network nodes (as written in ROC[\( v \)].MaxGain), then update the node’s gain field in ROC-PDU (at ROC[\( v \)].MaxGain) and send the ROC-PDU to the next node on the virtual ring.

7. If the gain of the added VP is less than a threshold \( \text{Gain}_{\text{min}} \), do not add any VP and send ROC-PDU to the next node on the virtual ring (and wait for the next event).

8. (Otherwise) add the description of the VP to ROC[\( v \)].NewVPs.

9. Update the local picture of the VP layout (and the routing of VCs with respect to it).

10. Start a VP setup protocol of the new VP.

11. Go to line 5, to find the next potential VP.
5. Upon completion of each such reroute, the delete pending VP, which is unused by now, is deleted using the standard takedown procedure [5].

**Observation 1.**

1. After phase 2, all add pending VPs have been added, and may be used by new VCs.

2. After phase 2, no new VCs are routed using delete-pending VPs. Therefore, after all short-term VCs have ceased to use them, and all long-term VCs have been rerouted to other VPs, these delete pending VPs may be deleted safely with no data loss.

After activation of the CNA algorithm, the network has adjusted to the new layout, and the entire process may restart periodically.

The main drawbacks of this centralized approach are:

1. The solution is vulnerable to crashes of the VPLM site, since no other site has the full picture of the network,

2. Large amounts of memory are required at the VPLM to hold network-wide data,

3. Large amounts of data exchange to and from the VPLM are necessary since every setup/takedown of VCs is reported to the VPLM, and topological updates are broadcasted from the VPLM to the rest of the network (as well as setup/takedown requests and confirmations).

### 3.2 A distributed approach

In this section we present a distributed version of the above algorithms (CA, DA, and CNA). This version is proven to produce the same sequence of changes in the VP layout as the centralized version, yet it does so with a low amount of information transfer and improved fault tolerance.

To achieve low data transfer, the knowledge at each node on the utilization pattern of the network, is restricted to data which may be acquired locally, during ordinary network procedures (e.g. topology updates, setup/takedown procedures etc.), implying no communication overhead for it: a node will have data only on VCs for which it serves as a VC switch but not on VCs for which it is just a VP switch, since at a VP switch of a VC, the network layer is not involved in the setup protocol and is thus not aware of the existence of the VC.

As far as VPs are concerned — each node stores the entire VP layout. This data must be present at each node for other reasons as well (e.g. for deciding on the route of a new VC that starts at the node) and thus is not considered an overhead.

Based on this local view of the network, each node decides on its best candidate for a new add-pending VP, and its best candidate for a delete-pending VP (if such candidates exist). However, the decision on the next add-pending VP must be done globally, by selecting the candidate with maximum gain (i.e. \( Gain(p_x) = \max_{e \in V} Gain(p_e) \)). To this end, the network layer modules at all nodes are interconnected by a virtual ring (using VCs), on which a "token" circulates. We term this token

---

8In the rest of this section we refer to VP addition only. Since VP deletion is analogous (as shown in the discussion on the centralized approach), we omit it for sake of brevity.
The algorithm assumes the existence of a static VP layout as part of the full VP layout. The VPs of this static VP layout are not candidates for VP deletion, thus ensuring that some lower bound on the performance of the network is preserved (i.e. that of the static VP layout).

Algorithm CD (Centralized Deletion):

0. Given a layout of VPs in the network, a layout of VCs and the routing of each VC (as in the CA algorithm).
1. Loop the next lines until the loss (see definition in Stage 2) incurred by the last deleted VP increases above some threshold \( \text{Loss}_{\text{max}} \).
2. For every VP in VP layout which is not part of the static VP layout, compute the increase in \( \mathcal{H} \) had the VP been removed from the VP layout (termed the LOSS of the VP); This is computed by multiplying the number of VCs that use the VP as part of their route by the length of the next shortest route in the VP layout which uses the same physical links, minus one.
3. Find a VP with minimal loss and mark it as "delete pending".
4. Remove the VP from the local topological picture at the VPLM.
5. Substitute the VP by the shortest sequence of VPs that replaces it, in the routing of all VCs that use it.

Note that the CD algorithm is dual to the CA algorithm, but here we select an existing VP with minimal loss (in terms of \( \mathcal{H} \)), rather than a potential VP with maximal gain (as in CA). To find the loss of a VP, we must first calculate the shortest alternative route, to be used by VCs instead of the deleted VP.

After having found a list of add-pending and delete-pending VPs, VPLM should adjust the rest of the network to the new VP layout, with low overhead. At this stage the add-pending VPs are actually constructed and the delete-pending VPs are destroyed, and these topological changes are distributed to the rest of the network by global updates.

Algorithm CNA (Centralized network adjustment):

0. This protocol is initiated by VPLM after the CA and DA algorithms have terminated, and proceeds in phases which are orchestrated by it.
1. The nodes that serve as VP endpoints to add-pending VPs, are notified to start a VP setup protocol (see [5] for details).
2. Upon completion of the VP setup protocols, the VP endpoints notify VPLM (which waits for all the notifications); VPLM then distributes the new topological database (including both add and delete pending VPs) to the nodes, so that new VCs use the improved VP layout,
3. Pairs of nodes that serve as endpoints to new VPs, are notified to perform in-path rerouting (IPRR, see Section 4 for the details) of all permanent (or long-term) VCs that include them, from the old VP layout to the new one.
4. Each pair of endpoints of delete-pending VPs, are informed to perform IPRR for all the VCs that use them, from this VP to the next shortest VP sequence (on the same physical route).
While a mathematical analysis of the approximation is very difficult, it is easy to see that all the added VPs do contribute to the reduction in $\mathcal{H}$ at least as we expected them (i.e. that an addition of a new VP does not decrease the contribution of VPs that were added previously):

**Proposition 1.** Let $\mathcal{H}$ be the total VP hops for a given VP layout, let $\mathcal{H}'$ be the total hops for the new VP layout, after the application of the CA algorithm on it. Assuming that $k$ VPs were added. Let $Gain_i$ denote the gain of the $i^{th}$ VP added by CA, at the time it was selected as a new VP. Then

$$\mathcal{H}' \leq \mathcal{H} - \sum_{i=1}^{k} Gain_i.$$ 

The following lemma addresses an important property of the problem, and is used in the sequel to prove that the distributed version of the algorithm produces identical results to that of the centralized version. The lemma states that the gain of adding a VP to the VP layout does not increase (assuming that the VC layout is fixed). Formally:

**Lemma 1.** The values of $Gain_i$ are monotonically decreasing (i.e. $i < j \Rightarrow Gain_i \geq Gain_j$).

**Proof.** We shall prove that an addition of a VP does not increase the gain of using any other potential VP. Using this fact, it follows that if an algorithm always chooses to add a VP with maximal gain, all the other potential VPs do not have a higher gain, and their gain may not increase as a result of this addition. From this the lemma follows directly.

The proof considers the relation between the added VP $a$ and a sequence of VPs, $b$, which form a potential VP to be added.

**Case 1:** The routes that $a$ and $b$ take in the network are disjoint. In this case the number of VCs that use $b$ is not influenced by the addition of $a$, and the gain of $b$ does not change.

**Case 2:** The VP $a$ is equal to a subsequence of $b$. In this case the number of VP hops that will be reduced from $\mathcal{H}$ if $b$ were added as a VP is reduced, and thus the gain of $b$ is reduced.

**Case 3:** Both ends of $b$ are endpoints of VPs which were part of $a$ (before it was added as a VP). In this case the utilization of $b$ is decreased, since VCs that may use $b$ will use $a$ instead (if they include the full route of $a$), and the gain of $b$ decreases.

**Case 4:** The routes of $a$ and $b$ share some part, but in a different way than mentioned in cases 2 and 3. It is easy to see that in this case the VCs that use $a$ and the VCs that use $b$ are disjoint, and the gain of $b$ does not change. $\square$

So far we have discussed the addition of new VPs to a layout. It is equally clear that the network management must delete VPs that become useless, or else the VP layout increases unnecessarily, and network management overheads are increased. The deletion algorithm finds VPs that may be deleted without increasing $\mathcal{H}$ substantially. A VP is selected for deletion (i.e. marked delete-pending) if $\mathcal{H}$ of the VP layout after the VP is removed is not increased by more than some threshold $Loss_{max}$. As in the CA algorithm, this algorithm does not alter the actual VP layout, but only marks the VPs as "delete-pending", in the local topological picture at VPLM, for later use by the network-adjustment phase.
The gain of adding VP4=(VP1,VP2,VP3) is 4*(3-1)=8

Algorithm CA (Centralized VP Addition):

0. Given a layout of VPs in the network, a layout of VCs and the routing of each VC as a sequence of the VPs.
1. Loop the next lines until the gain of adding a new VP is lower than some threshold Gain\_min (see definition of gain in Stage 3): 
2. Find sequences of VPs which are used by some of the VCs in the VC layout. These sequences are candidates for addition as new VPs.
3. Compute the amount by which $\mathcal{H}$ is reduced, had each of the candidates been added to the VP layout (termed the GAIN of using the candidate); This is done by multiplying the number of VCs that use the sequence (and would thus use the candidate VP, had it been added) by the length of the sequence, minus one.
4. Choose the candidate which maximizes the gain, as a new VP and mark it as "add pending" (see Figure 2).
5. Add the candidate to the topological picture which is maintained by the VPLM.
6. Update the routing of VCs that use the sequence, to use the new VP instead (again, only the local picture at the VPLM is updated at this stage).

The CA algorithm is a steepest descent method, since it chooses the next approximation along the steepest slope (i.e. the maximum gain) of the target function $\mathcal{H}$. Such methods are often used as a heuristic in network design, and are known to produce good (although not necessarily optimal) results.

\(^2\) Note that this is an update in the local topological picture at the VPLM. The new VP will be set up and the new topology distributed at a later stage to decrease the rate of updates in the network.
3 The solution

In this section we propose two versions for a solution to the problem. We first present a centralized version of the solution, and then present a distributed version which yields the same results. Both solutions are based on three main parts:

1. Finding potential VPs, the addition of which substantially reduces $\mathcal{H}$ (we term these VPs \textit{adding VPs}),

2. Finding VPs that can be deleted without substantially increasing $\mathcal{H}$ (\textit{deleting} VPs),

3. Adjusting the network to the new layout, including: (1) Setting up the new, adding VPs; (2) Distributing the new topological data, so that new VCs will be routed using these VPs; (3) Rerouting long-term VCs that use the "old" VP layout, according to the improved layout; (4) Tearing down the deleting VPs (which are by now not used by any VC).

3.1 A centralized approach

The centralized approach is based on a VP layout management center in the network (VPLM for short), to which all events concerning the setup and takedown of VPs and VCs are reported. These events are used by VPLM to deduce statistics on the utilization pattern (i.e., the VC layout) of the network. The VPLM keeps a full topological picture of the network and the routes of the current VPs and VCs. Periodically, the VPLM checks if the VP layout should be changed to better suite the current utilization of the network. This is done by activating VP addition and deletion algorithms (CA and DA, described in the sequel) at the VPLM. If such changes improve the VP layout, a network adjustment algorithm is activated, to change the VP layout in the network "smoothly" (see the CNA algorithm in the sequel).

We first present the VP addition algorithm, which is a greedy (steepest descent) algorithm: At every iteration, a sequences of VPs which is used by many VCs is joined together to form a new VP. This VP will be used for future VCs instead of the sequence (see Figure 2 for a visual example).

![Figure 1: A layout of VCs and VPs on a network](image)

<table>
<thead>
<tr>
<th>VC</th>
<th>Routing in VP layout 1</th>
<th>Routing in VP layout 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>(A,B,C)</td>
<td>(A)</td>
</tr>
<tr>
<td>b</td>
<td>(E,D)</td>
<td>(C,B)</td>
</tr>
<tr>
<td>c</td>
<td>(E)</td>
<td>(C)</td>
</tr>
<tr>
<td>d,e</td>
<td>(F,G)</td>
<td>(D)</td>
</tr>
</tbody>
</table>

$\mathcal{H} = 3 + 2 + 1 + 2 + 2 = 10 \quad 1 + 2 + 1 + 1 + 1 = 6$
2 Problem definition

For a given ATM network, we shall use the following definitions and notations.

Definition 1. A VC layout is a set of VC connections and the description of their paths in the network. A VP layout is the set of VP connections in the network.

Our goal is to find a VP layout that enables us to route each VC from a given VC layout along its preset route (determined by the VC layout), using only VPs from the VP layout. These concepts are clarified by the following definitions.

Definition 2 (Routing). A routing of a VC layout with respect to a VP layout specifies for each VC in the VC layout, a sequence of VPs from the VP layout, along which the VC is routed (such that the concatenation of the routes of these VPs is identical to the route of the VC).

Definition 3 (VP hop count). The VP hop of a VC with respect to a routing is the number of VPs along which the VC is routed. Let $\mathcal{H}$ denote the total number of VP hop counts over all the VCs $^1$.

Obviously there are many routings for a fixed VC layout and VP layout (see Figure 1); An optimal routing is a routing in which $\mathcal{H}$ is minimal.

Definition 4 (Optimal VP layout). Given a VC layout, an optimal VP layout is a VP layout with a small number of VPs, for which the optimal routing with respect to the VC layout yields a minimal $\mathcal{H}$.

Definition 5. We shall use the following terminology:

- A switch is a VC switch with respect to a VC, if it is an endpoint of a VP which is used by the VC (as determined by the routing). In such a switch, cells that belong to the VC are switched using the VCI.
- A switch is a VP switch with respect to a VC, if it is not a VC switch. In such a switch, cells that belong to the VC are switched using the VPI only.
- The utilization of a VP is the number of VCs from the VC layout which include the VP in their route (as determined by the routing).

Example 1. Consider the VC layout in Figure 1. It contains 5 VCs and their routes in a given network. The figure also contains two alternative VP layouts. The first VP layout has total hops $\mathcal{H} = 10$ (see the table), while the second VP layout has $\mathcal{H} = 6$. Since the second layout is also composed of a smaller number of VPs, it is considered a better layout.

In the sequel, we propose algorithms for decreasing $\mathcal{H}$ by adjusting the VP layout to dynamic changes in the utilization of the network (i.e. in the VC layout), while not increasing the number of VPs too much.

\footnote{This number clearly determines the average VP hop count for any VC.}
at a switch in which the VPI marks the end of the VP, is the VCI read in order to route the cell into another VP. Thus, as far as routing is concerned, the network may be viewed as a collection of VP routes (which we term the VP layout), and each VC connection may be viewed as a route which is composed of a concatenation of VPs.

The number of VPs used by a VC connection (termed VP hop count), is known to directly affect the efficiency of its setup, since the setup must update VC routing tables at a number of switches which is proportional to the VP hop count, an operation which involves the network layer at each such switch. For this reason and others, it is important to keep the number of VP hops of every VC as low as possible [2, 10]. While the straightforward solution of a direct VP between the endpoints of every plausible route is optimal for small networks, it is not practical for larger ones, since the total number of VPs is equal to the number of plausible VC routes, which may be very large and complicate global management tasks.

This phenomenon raises a new design problem of finding a VP layout which is composed of a small number of VPs, while keeping the VP hop count for any desired VC from increasing too much. In [3, 7] it is shown how to construct a static layout of VPs, so that there is an efficient route with low VP hop count between any pair of switches. However, this technique does not take the actual usage pattern of the network into account, and limits the possible routes for VCs.

In the present work we address the problem of adjusting the layout of virtual paths in an ATM network to the utilization pattern of the network. We discuss a dynamically changing VP layout, which learns the paths along which many VCs are created, and decreases the number of VP hop count for such paths, thus improving the average number of VP hops per VC (whereas in [3, 7] the layout was static, and catered for the worst case only).

Note that if these two techniques are combined, both the worst case VP hop count and the average VP hop count can be kept low.

In [1, 8] similar problems are presented, in which a static design of the VP layout is proposed, based on a large number of parameters. In these works, a set of routes for VCs is first found, which is then used as a basis for designing the layout of VPs. The algorithms presented there are based on standard optimization techniques, and are thus time-consuming. In contrast, we assume that the desired routes for VCs are given by some routing entity (based on complex considerations, such as bandwidth allocation), and focus only on the design of the VP layout, so that these VCs may be routed with a low number of VP hops. This approach enables us to design a low overhead algorithm which is suitable for working during the on-going activity of the network, and automatically adjusts the network to a changing VP layout without disrupting its ordinary operation (whereas [1, 8] are based on time consuming optimization methods, suited for an offline design of the layout).

In Section 2 we define basic concepts and problems of this paper; In Section 3 we discuss centralized and distributed algorithms that (1) find the needed changes in the VP layout so that it better fits the existing VCs; (2) adjust the network to the new VP layout with low overhead and without disruption to the network. In Section 4 we show a new rerouting protocol which enables to change the VP layout "smoothly" and is used by the above-mentioned algorithms. In Section 5 we discuss bandwidth allocation aspects of our solution, and summarize the results in Section 6. The formal notation for the problem, along with formal definitions and algorithms are found in the Appendix.
Dynamic Maintenance of the Virtual Path Layout

Ornan Gerstel and Adrian Segall
Computer Science Department
Technion, Israel Institute of Technology
Haifa 32000, Israel
E-mail: orig@csc.cs.technion.ac.il, segall@cs.technion.ac.il

Abstract
In this paper we discuss methods for adjusting the layout of virtual paths in an ATM network, to the dynamics of changes in the usage of the network by its end-users. We first present a centralized algorithm for finding a better layout for the current traffic pattern, and for applying the change in the network and discuss its drawbacks. We then present a distributed algorithm that emulates the centralized algorithm with a much lower overhead, and enhanced durability to faults. We prove that both algorithms produce identical layouts, thus showing the superiority of the latter algorithm. Both algorithms base the changes in the network on a new rerouting protocol, which does not cause any losses in data, nor changes in the FIFO order of cells.

Keywords: ATM, Virtual Paths, VP Routing, Network Management, Distributed Algorithms, Rerouting.

1 Introduction
The Asynchronous Transfer Mode (ATM) is the transmission and multiplexing technique chosen by ITU for B-ISDN and by a large number of fast network designers and vendors (e.g. [9, 6]). ATM provides a connection-oriented service via virtual circuits, and is based on the switching of packets (called cells), using VP/VC routing labels at the cell’s header (denoted VPI/VCI), and VP/VC routing tables at the switches of the network. The ATM standard specifies two types of connections — Virtual Path connections (VPs) and Virtual Channel connections (VCs). While VCs are used by network users as the virtual circuits on which data is transferred, VPs are used to bundle several VCs together, thus decreasing the amount of entities to be managed.

This functionality is realized by a hierarchical scheme, in which the VCI of a cell is ignored in many switches along the route traversed by it, where routing is done according to the VPI exclusively. Only