A Flexible Rerouting Protocol in ATM networks*

Yael Dubinsky† and Adrian Segall‡

July 14, 1998

Abstract

The paper introduces a protocol for rerouting calls in an ATM network, referred to as the Flexible Rerouting Protocol (FRP). Failure of a VP triggers construction of an alternate VP out of VP's that are not in use, named stand-by VP's, by simple operations at the endpoints of the stand-by VP's. VC rerouting from the failed VP to the alternate VP is performed concurrently. FRP is designed to also allow recovery from repeated failures that may occur during the construction and rerouting process.

1 Introduction

The Asynchronous Transfer Mode (ATM) is a switching and multiplexing technique selected to be used in Broadband ISDN (B-ISDN) networks [4, 7, 8, 13]. A two level hierarchy of ATM connections is defined and is known as "the virtual path concept". This concept defines Virtual Paths (VPs) which are predefined circuits composed of one or more concatenated transmission links, and Virtual Channels (VCs) which define connections between end users [1, 15, 16, 17]. In Figure 1, an end user connected to switch A can communicate with an end user connected to switch F via a VC that traverses two VPs, between switches A and D and between switches D and F respectively.

Failures of switches or transmission links of a VP annul the VP and without a rerouting protocol, lead to the cancellation of all VCs that use the VP. Thus efficient VC rerouting protocols to alternate paths add considerably to the reliability of ATM connections. There are two main approaches to constructing an alternate VP upon failure of a VP that is being used by VCs [3, 6, 9, 12, 11]: either maintain a backup VP for every active VP or construct one from scratch when the failure occurs.

In [3], three rerouting algorithms are examined. The first is the local rerouting algorithm that reroutes around the failed link, without regard to source/destination switches. The second is the

---

*This paper is based on a Technion M.Sc. Thesis
†Dept. of Computer Science, Technion, Haifa, Israel, yael@cs.technion.ac.il
‡Dept. of Electrical Engineering, Technion, Haifa, Israel, segall@ee.technion.ac.il
source based rerouting algorithm, in which the source reroutes to a pre-computed alternate path. The third is the local destination rerouting algorithm, where the local switch reroutes to the best alternate route from itself to the VP destination. The results of [3] show that source based rerouting is superior in terms of necessary redundant capacity, especially when multiple links fail.

In [9], when a failure occurs, the endpoint switches of the failed VP broadcast search messages to their neighbors. Each switch that receives a search message reserves capacity, increments the hop number in the message and broadcasts to its neighbors. When a switch receives search messages from both endpoint switches, it combines the information in the messages and sends it to the endpoint switches. When an endpoint switch receives information about the alternate route, it sends an acknowledge message over this route in order to activate it.

The protocol for rerouting VC’s from a failed VP to a backup VP is given in [6]. It is indicated in this work how the rerouting protocol reroutes active VC connections to a backup VP and how the rerouting protocol enables other signaling protocols (i.e. the VC setup and takedown protocols) to recover from loss of connection signaling messages. Moreover, if the VP failure occurs while a VC is in the process of being established, the protocol ensures that the setup procedure is completed over the backup VP. Finally, the paper shows how VC connections that cannot be accommodated by the backup VP, and therefore cannot be rerouted, are taken down.

In [12], a backup VP is maintained for every VP in the network. When a failure occurs, one of the endpoint switches, the Sender, switches the failed VP to the backup VP and sends a restoration message that saves the capacity along the backup route. When the other endpoint switch, the Chooser, receives the message, it switches the failed VP to the backup VP. If the backup VP is not available, a new restoration route is found dynamically by the endpoint switches that broadcast messages of the backup VP reconstruction algorithm. Each switch that receives the message, searches for backup routes and rebroadcasts the message. When the new routes are discovered, the shortest is selected and a connection message is sent over it that builds the new backup VP.

Another paper [11] studies topological issues related to backup VP assignment and maintenance, including second generation ones. It is proposed to assign a backup VP to each primary VP at system initialization as well as when need arises, using topological and capacity decisions. When a
failure occurs on the primary VP, the backup VP is used and operations are performed to maintain a stable layout of backup VPs. These operations are relocation of injured backup VPs that also use the failed link or switch and establishing a second-generation backup VP for each activated backup VP. There are periodical checks to discover topological changes that allow establishing backup VP's where previously no such VP's existed.

The main drawback of the backup VP approach is that the network designer has to maintain at all times a backup VP in parallel to every active VP. Its main advantage is its speed of rerouting, thus allowing more VC's to be rerouted before the application timer expires. The other extreme, constructing a VP from scratch upon failure, is slow but releases the network management protocols from ensuring a pre-established backup VP at all times.

The present paper introduces an ATM rerouting protocol, called the Flexible Rerouting Protocol (FRP), that approaches the recovery speed of the backup VP method, without its management restrictions. FRP expands on the idea of backup VP's and makes the system more flexible. Instead of insisting on a backup VP in parallel to every primary VP, we maintain a layout of stand-by VP's that cover the entire network in a mesh topology. A stand-by VP is simply a VP that is not currently in use by any VC. Design of an optimal layout of the VP's, active and stand-by, is not within the scope of this paper. The more stand-by VP's are maintained, the easier is for the endpoints of a failed VP to construct an alternate one. Whenever an active VP fails, one of its endpoints, designated as the primary endpoint, searches for one or more routes to the other endpoint, each such route consisting of one stand-by VP or of a concatenation of stand-by VP's. By simple operations performed by the endpoints of the concatenated stand-by VP's, a new VP is constructed and the VC's that have been using the failed VP are rerouted.

Obviously, the backup VP protocol is a special case of FRP, since an alternate route consisting of a single stand-by VP is in fact a backup VP. The generalization from a backup VP to a concatenation of stand-by VP's provided by FRP gives the advantage of reduced VP management constraints. There is no need anymore to make sure that every active VP has at all times a backup VP. Therefore the VP management facility does not need to react as fast as before and can take more time to optimize stand-by VP layout.

We have designed FRP to allow for recovery from repeated failures. In particular, if there are additional failures during the construction of the new VP or during rerouting, FRP provides the mechanism for repeated attempts to find alternate routes and to reroute VC's over these routes.

Finally, the design of FRP reduces the size of control messages sent during reconstruction and rerouting by employing blocks of entries in the table of VC's maintained at each VC switch, one block per VP that currently terminates at the VC switch. The savings in communication achieved by this method will be explained in Sec. 3.2.

One of the important questions related to FRP is the design of a good layout for VP's and in
particular of stand-by VP’s. VP layout design has been a topic of ongoing research [2, 10, 5] and many open questions still exist. In the present paper we concentrate on the design of the protocols for building the new VP as a concatenation of stand-by VP’s in FRP, for rerouting VC’s on the new VP and for recovery from repeated failures during construction and rerouting. We provide a verbal description of the protocol, the pseudo-code for implementing the protocol and the validation proof of its properties.

The rest of the paper is organized as follows: The system model is defined in Section 2. In Section 3 we describe FRP and provide its pseudo-code. The properties of the protocol are presented and proved in Section 4. In Section 5 we present simulation and latency results and conclude in Section 6.

2 Model Description

2.1 Components

The system is composed of switching nodes connected by transmission links. A VP that is defined from switch A to switch E, is denoted by VP(\(AE\)). We assume that both VP(\(AE\)) and VP(\(EA\)) use the same path and denote the pair by VP(\(AE\)). Switches A and E are the endpoint switches of VP(\(AE\)). All other switches of VP(\(AE\)) are intermediate switches for this VP. A VP is said to be in-use if there is at least one VC that uses it. A VP that is not in-use is referred to as a stand-by VP.

In ATM, traffic is carried on Virtual Channels (VC’s) by cells that are routed by label swapping. The label consists of two fields in every cell header, the Virtual Path Identifier (VPI) and the Virtual Channel Identifier (VC1). When a cell traverses a VP, only the VPI is used. When a cell arrives at the endpoint switch of a VP, the VCI field is also used in order to route the cell into a new VP. The ATM Forum [1] distinguishes between two types of switches. A VC switch serves either as an intermediate switch or as an endpoint switch of a VP. Such a switch, when used as an endpoint switch, uses both the VPI and the VCI fields of a cell for routing. The second type can only serve as an intermediate switch in a VP and is referred to as a VP switch. Such a switch can use only the VPI field of a cell for routing.

Switches maintain tables that are used for swapping the VPI and VCI values of traversing cells. Both types of switches maintain a VP-Table (VPT) per port. Each entry in this table consists of two fields: the VPI field and the Link field. Every entry in the VPT represents a VP in which the switch serves as an intermediate or endpoint switch. The second table, the VC-Table (VCT), is maintained by VC switches only, one table per port. An entry in this table consists of three fields, the VPI, the VCI and the Link. VCT entries are organized in blocks, where each block corresponds to a given in-use VP that terminates at this port.
In addition to the tables, a switch maintains a processor in each port that handles the cells arriving at the port. The port processor in switch A in the port leading to switch B is denoted by $A_B$. The VP and VC tables in this port are denoted by $\text{VPT}(A_B)$ and $\text{VCT}(A_B)$ respectively. The link from A to B will also be denoted by $A_B$.

The Signaling ATM Adaptation Layer (SAAL)[14] enables signaling communication between ATM switches. On every VP($AE$) there is a VC, denoted by $\text{SVC}(AE)$, carrying signaling information for that VP. Two neighboring switches are always capable of transferring signaling information on the physical link that connects them, whether the link is part of a VP or not.

In order to identify a VP or a VC uniquely in the system, we assume that there is a global identifier for each, denoted by $\text{VPid}$ and $\text{VCid}$ respectively. These identifiers will be used by the signaling messages of the proposed rerouting schemes. VPs are established by a VP setup protocol [1], that is part of the management control of the system and contains a registration procedure that updates the topology databases in the network.

We assume that every VC switch maintains or has access to a topology database of the entire network, that contains the current physical topology, the currently available bandwidth resources, the VP layout and the VP usage. Using this database, a VC switch can find out if there is a route of stand-by VPs from itself to any other VC switch.

A VC switch maintains an information table for every VC that it serves. The information consists of the id of the VC, denoted by $\text{VCid}$ and a rerouting counter, denoted by $\text{VCcounter}$. The counter is initialized to zero when the VC is set, and is canceled when the VC is canceled. The use of the rerouting counter is described in Section 3.

### 2.2 Routing by label swapping

The label swapping in ATM is well known and will only be summarized here for convenience of later referral.

#### 2.2.1 Label swapping in a VP

We show in Figure 2 the label swapping mechanism on VP($AD$) and VP($DA$) of Figure 1. Consider a cell on a VC that uses VP($AD$). The cell enters the VP, i.e. is inserted by switch A into the link $A_B$ to B with some label $b_1$. The pair $(b_1, A_B)$ is defined as the starting label of VP($AD$). When switch B receives the cell, it swaps label $b_1$ with label $c_1$, according to the entry $b_1$ in $\text{VPT}(B_A)$ and routes the cell on link $B_C$. The process is repeated until the cell arrives at switch D, that is an endpoint switch of VP($AD$). Switch D finds the value nil in the VPI field of entry $d_1$ in $\text{VPT}(D_C)$, that that the cell has arrived at the end of VP($AD$). Similarly, a cell traveling on
Tables of VP(DA):

<table>
<thead>
<tr>
<th>VPI</th>
<th>Link</th>
<th>VCT block</th>
</tr>
</thead>
<tbody>
<tr>
<td>nil</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a_2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tables of VP(AD):

<table>
<thead>
<tr>
<th>VPI</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_1</td>
<td>B_C</td>
</tr>
<tr>
<td>c_1</td>
<td>D_C</td>
</tr>
</tbody>
</table>

Figure 2: Label swapping using VPTs

VP(DA) uses (c_1, D_C) as the *starting label*, then VPI=b_2 when forwarded by switch C, VPI=a_2 when forwarded by switch B and VPI=nil at switch A. A formal definition of a VP follows:

**Definition 1** A unidirectional VP is a starting label and a collection of entries in VP tables at switches along a unidirectional path, with the following properties:

(a) Each entry has a label, e.g. b_1 in VPT(B_A) (all examples refer to Figure 2).

(b) The starting label and each entry contain two fields: VPI and Link. The VPI field is the label of the entry at the next switch in the path, e.g. c_1 in entry b_1 in VPT(B_A). The Link field is the port that leads to the next entry, e.g. B_C in entry b_1 in VPT(B_A).

(c) The last entry in the path is referred to as the VP ending label, e.g. entry d_1 in VPT(D_C). Its VPI field is nil and its Link field points to the VCT block corresponding to the VP.

**Definition 2** A VP is a pair of unidirectional VPs connecting two switches in opposite directions, along the same path.

2.2.2 Label swapping in a VC

Each switch maintains VCT's of VC entries, one table per port. In this paper we assume that each VCT is divided into blocks, one block for every in-use VP that terminates at this switch at this
port. The entries of a block are contiguous and each represents a VC that uses the VP of this block. When a VC cell arrives at an endpoint switch of a given VP, its VCI is used as an offset in the VCT block of this VP. This is done by masking the VCI in the cell with the starting address of the block in a way that keeps the starting address in the Most Significant Bits and the offset in the Least Significant Bits. For example, suppose each block has 1024 entries. The VCT entry referred to by a cell is determined by masking the ten LSBs of the starting address of the block with the appropriate bits in the VCI of the cell. Using VCT blocks reduces the number of VP’s and VC’s per switch per port. If each VCT block has 1024 entries, only $2^8$ VP’s can traverse this port and each VP can handle up to 1024 VC’s. Using VCT blocks requires a mask operation every time there is an access to a VCT entry. However, the advantage of using VCT blocks is that when rerouting over a chain of VPs, there is no need for the endpoint switches of the failed VP to agree on a new VCI for every VC that is rerouted. This will become apparent in Section 3.

Consider VP($AD$) of Figure 1 that ends at switch D. Switch D receives cells of VP($AD$) at port $D_C$ leading to switch C, thus VCT($D_C$) contains a block of entries for this VP, as shown in Figure 3. The entry $ad_i$ in this block belongs to a VC that uses VCI=$ad_i$ on the VP from switch A to switch D. Consider a VC between switches A and F using VP($AD$) and VP($DF$), as depicted in Figure 1. When a cell of this VC arrives at switch D, its VPI field is $d_1$ as shown in Figure 2. Since the VPI field of entry $d_1$ in VPT($D_C$) is nil, switch D follows the Link field to the block of VP($AD$) in VCT($D_C$) using the VCI field of the cell. Switch D updates the cell with new VPI and VCI values that are found in entry $ad_i$ in VCT($D_C$), and forwards the cell to Link $D_E$ on VP($DF$). Here $D_E$ is the value in the Link field of entry $ad_i$ in VCT($D_C$). When the cell arrives at the last endpoint switch of the VC, it finds nil in the VPI field of the appropriate VCT entry. This nil value signals the end of the VC and the cell is delivered to the appropriate AAL process.

We shall need in the sequel a formal definition of a VC:

**Definition 3** A unidirectional VC is a starting label and a collection of entries in VC tables at VP endpoint switches along a unidirectional path, with the following properties:

(a) Each entry has a label, e.g. $ad_i$ in VCT($D_C$) (all examples refer to Figure 3).
(b) The starting label and each entry contains three fields: VPI, VCI and Link. The VPI and Link fields are the starting label of the next VP in the path, e.g. \(d_{10}\) and \(D_E\) in entry \(a_d\) in \(VCT(D_C)\). The VCI field is the label of the entry at the next endpoint switch in the VC path, e.g. \(d_f\) in entry \(a_d\) in \(VCT(D_C)\).

(c) The VPI field of the last entry in the path, denoted by VC ending label, is nil. The Link field points to the appropriate AAL process.

**Definition 4** A VC is a pair of unidirectional VCs connecting two switches in opposite directions, along the same path.

**Definition 5** An endpoint switch of a VP can be in one of two states for each unidirectional VC that uses the VP: active or passive. In state active it forwards cells of this VC according to the table entries and in state passive it discards those cells. We say that the VC is ready at a switch if both its unidirectional VCs are in active state at that switch.

### 2.3 Stand-by VPs

As defined before, VP’s that are not in use by any VC are referred to as stand-by VP’s. There is no VCT block corresponding to a stand-by VP. The Flexible Rerouting Protocol (FRP) is a protocol for best-effort rerouting of calls in case of link/switch failure. The protocol uses concatenations of stand-by VP’s to build a new VP, denoted by \(NewVP\), over which the calls will be rerouted. The endpoints of the stand-by VP’s that are to be concatenated into \(NewVP\) will be referred to as the Main Intermediate Switches of \(NewVP\).

### 3 The Rerouting Protocol

We say that a VP fails if one of its transmission links or intermediate switches fails. The VCs that have been using a failed VP must either be cancelled or rerouted. The purpose of the present work is to suggest a flexible protocol for rerouting such VC’s. The endpoint switches of the failed VP learn about the failure and activate the rerouting protocol. In order to break symmetry, we shall designate the endpoint that has established the VP as the one that controls the rerouting procedure and shall refer to it as the rerouting endpoint. The other endpoint switch will be referred to as the non-rerouting endpoint.

Whenever a failure occurs on a link or at a switch that is used by a VP, the switches adjacent to the failure generate a FAIL message that propagates between adjacent switches towards the
endpoints of the VP. The FAIL message contains the location of the failure. The rerouting protocol begins when the endpoints receive the FAIL message. The non-rerouting endpoint puts all VC’s that use the considered VP in state passive and waits for a message from the rerouting endpoint. Cells that arrive on a VC in state passive are discarded. Each endpoint switch uses a timer that is initiated for each VC that leaves ready state. If the timer expires before the VC is routed, the VC is cancelled. When the rerouting endpoint receives the FAIL message, it searches for an alternate route that avoids the location of the failure (see Fig. 4), performs VCT operations and sends a REROUTE message on the alternate route. The REROUTE message triggers the construction of the new VP, denoted by NewVP, at the Main Intermediate Switches and the VCT operations at the non-rerouting endpoint. The latter decides which VC’s to reroute according to the value of VCcounter and responds to the rerouting endpoint with a REROUTE-ACK message. The rerouting endpoint repeats its attempts to reroute VC’s as long as there are VC’s that have not been cancelled and that have not been rerouted yet. Details of the protocol are given in the following subsections.

![Diagram](image)

Figure 4: Rerouting over a bypass

3.1 Operations on the VP and VC tables

Before proceeding, we define three basic operations on the VP and the VC tables that are performed by the rerouting protocol.

We shall illustrate those operations on a VC as in Figure 5 between switches W and Z that uses VP(AE). Suppose that cells of VC(WZ) have VCI=w1 when arriving at switch A from switch Y. Before rerouting, entry w1 contains (b1,a3,A). A.

Suppose that switch A wants to reroute cells of VC(WZ) to another VP that starts at switch A, goes through switch L with starting label l1, and ends at switch E. Switch E will use the same VCI label for cells sent over the new VP as on the old one and therefore the modifications that switch A must perform in VCT(A) are as follows: The outgoing link is changed to A, the VPI value is changed from b1 to l1 and the VCI value is not changed. This operation, as shown in Figure
5, is denoted by $VCT\text{-}update(A_Y)$.

Consider now VC($ZW$) that uses VP($EA$) and suppose that cells belonging to this VC carry VCl=$ea_1$ when arriving at switch A from switch B. Entry $ea_1$ in VCT($A_B$) contains $(y_1, aw_4, A_Y)$.

After rerouting the VC, switch A should be able to receive cells of VC($ZW$) on the VP that starts at switch E, goes through switch L, and ends at switch A. Therefore, switch A has to move the $ea_1$ entry from VCT($A_B$) to the block of entries in VCT($A_L$) that corresponds to the new VP. After this operation, entry $ea_1$ does not exist in VCT($A_B$). This operation, as shown in Figure 5, is denoted by $VCT\text{-}move(A_B, A_L)$. A $VCT\text{-}move$ operation may be performed on blocks of different VCTs, as explained above or on blocks in the same VCT. The latter occurs when the new VP uses the same port as the failed one. Rerouting is performed only to stand-by VPs and therefore entry $ea_1$ in the block that corresponds to the new VP, and in fact the entire block, is available.

The third type of basic operation is $VPT\text{-}concat$. Its purpose is to concatenate two stand-by VPs into one.

Consider Figure 6. In this Figure there are two bi-directional VPs, VP($AR$) and VP($RE$). We now want to concatenate the stand-by VP($ER$) with VP($RA$).

The entry of VP($ER$) in its last switch R is $r_{11}$ in VPT($R_G$), and the starting label of VP($RA$) is $h_{12}$ in VPT($R_H$). Thus, the $VPT\text{-}concat$ operation consists of writing the values $h_{12}$ and $R_H$ into the VPI and Link fields of the $r_{11}$ entry respectively. Concatenation of the stand-by VP channel
of VP(AR) to that of VP(RE) is performed in a similar way. By concatenating the two pairs of the stand-by VP channels, an operation denoted by VPT-concat(R_H,R_G), we are building a new bi-directional connection at the VP level between switches A and E that can be used as a new VP or as part of a new VP.

We now proceed to describe the rerouting actions that use the VCT-update, VCT-move and VPT-concat operations.

3.2 The protocol

The rerouting endpoint initiates the rerouting procedure when it receives a FAIL message on an in-use VP. At this time it enters Rerouting Mode, where it looks for alternate routes, named bypasses, consisting of concatenations of stand-by VP’s over which the VC’s can be rerouted. The rerouting procedure consists of building a new VP on each of the alternate routes and of rerouting VC’s from the failed VP to the new VP. Figure 8 shows a bypass between switches A and E with one main intermediate switch R. We denote this chain by A-R-E and the VP that is being built by NewVP(AE). Switches A and E activate a timer for each VC that moves out of ready state and uses VP(AE). If the timer expires before the VC is rerouted, the VC is cancelled. We use the VPT-concat operation in each of the main intermediate switches in order to concatenate the stand-
by VP’s on the bypass into one new VP and the \textit{VCT-update} and \textit{VCT-move} operations in the endpoint switches in order to forward the cells to and from the new bypass.

A VC that has not been rerouted yet and has not been cancelled because of timer expiration is in one of two lists at the rerouting endpoint. The \textit{Waiting} list contains the VClid’s of the VCs that have not been rerouted yet and for which there is no current rerouting attempt. The \textit{Attempting} list contains the VClid’s of the VCs that have not been rerouted yet and for which rerouting is being attempted. The transitions between the lists are illustrated in Fig. 7.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig7.png}
\caption{The VC states and transitions in the rerouting endpoint}
\end{figure}

We illustrate the rerouting procedure in Fig. 8. When switch E, the rerouting endpoint, receives the FAIL message, it activates \textit{timer}(VC) for all VC’s that have been in ready state, moves all VC’s that have been using the failed VP to \textit{passive} state and puts them in the \textit{Waiting} list. If a bypass A-R-E is found, switch E creates a VCT block for NewVP(\texttt{AE}) in port \texttt{E}_F, and performs \textit{VCT-update}(\texttt{EX}) and \textit{VCT-move}(\texttt{ED}, \texttt{EF}) for those VC’s in the \textit{Waiting} list for which rerouting to NewVP will be attempted. Because of resource restrictions, it may be the case that not all VC’s in \textit{Waiting} can be rerouted to the same NewVP. Switch E moves those VC’s to the \textit{Attempting} list, increments the VCcounter for each VC and includes the VClid and VCcounter of each VC in the INFO array. Then switch E sends a REROUTE message to switch R on SVC(\texttt{ER}) that contains NewVPId(\texttt{AE}), the topology of the bypass, loosely denoted by NewVP(\texttt{AE}), and the INFO array. Switch E moves the unidirectional VC’s that use VP(\texttt{AE}), to \textit{active} state for those VC’s. Since switch E can use in the VCT of the new block the same VCI entries as before, there is no need to include the VCI’s in the REROUTE message. Switch E keeps searching for bypasses and rerouting VC’s as long as the \textit{Waiting} list is not empty. Upon receiving the REROUTE message, switch R performs VPT-concat(\texttt{RH}, \texttt{RG}), that concatenates the stand-by VP(\texttt{ER}) and VP(\texttt{RA}) and sends a REROUTE message to switch A on SVC(\texttt{RA}).

Upon receiving the REROUTE message, switch A creates a VCT block for NewVP(\texttt{AE}) and compares the value of VCcounter in the message with $\text{VCcounter}_A$, the value for this VC stored
at A for each received VCid in the INFO array. If the latter is larger, it means that switch A has previously received a rerouting request for that VC that was sent later than the current one and thus switch A rejects the current request. Otherwise, it accepts the request by updating \( VCT_{counter,A} \), moving the VC to passive state and performing \( VCT\text{-}update(A_Y) \) and \( VCT\text{-}move(A_B,A_L) \) for that VC. The REROUTE-ACK message sent back by A to E on NewSVC(AE) contains an INFO array consisting of the accepted VC’s. At this point the rerouting procedure is completed at switch A for these VC’s, i.e. A moves the VC’s that have been rerouted to \textit{ready} state and resumes sending their cells, on NewVP(AE).

Upon receiving the REROUTE-ACK message, switch E can determine which VC’s have been accepted or rejected at A. The first kind are \textit{ready}, the second kind return to the \textit{Waiting} list. If the \textit{Waiting} list is not empty, further searches for alternate routes are undertaken by E.

Next we present the details of the rerouting procedure, as applied to a specific VC(\( WZ \)), that uses three VP’s: VP(\( WA \)), VP(\( AE \)) and VP(\( EZ \)), as depicted in Figure 8(a). We describe the changes in the VPTs of switch R and in the VCTs of switches E and A which are triggered by the \textit{rerouting procedure}.

Cells of VC(\( WZ \)) are transmitted from switch Y to switch A with VCI=\( wa_1 \), from switch A to switch E with VCI=\( ae_3 \) and from switch E to switch X with VCI=\( e_2z_1 \). The VC in the opposite direction VC(\( ZW \)) uses respectively VCI=\( ze_5 \), VCI=\( e_5a_1 \) and VCI=\( aw_4 \).

Switch E performs \textit{VCT\text{-}update} in order to reroute VC(\( ZW \)) from VP(\( EA \)) to NewVP(\( EA \)), and a \textit{VCT\text{-}move} operation in order to reroute VC(\( WZ \)) from VP(\( AE \)) to NewVP(\( AE \)). As shown in Figure 8(b), the \textit{VCT\text{-}update(E_X)} operation updates entry \( ze_5 \) with \( f_2 \), the \textit{starting label} of stand-by VP(\( ER \)) and with \( E_F \), the new Link. The \textit{VCT\text{-}move} operation moves entry \( ae_3 \) from VCT(\( E_D \)) to VCT(\( E_F \)). The VCI entries in VCT(\( E_F \)) for the VCs that will use NewVP(\( AE \)) are all empty before this operation because NewVP(\( AE \)) is new at port \( E_F \). After performing these operations, switch E moves VC(\( WZ \)) to \textit{active} state.

Then, switch E sends a REROUTE message on SVC(\( ER \)) that contains NewVPid(\( AE \)), the bypass topology and the couple VCid(\( WZ \)), VCCounter(\( WZ \)). Upon receiving the REROUTE message, switch R performs \textit{VPT\text{-}concat}(\( R_H,R_G \)) operation as shown in Figure 8(c); details of this operation were given in Figure 6. Switch R then forwards the REROUTE message to switch A on SVC(\( RA \)).

The operations performed by switch A for VC(\( ZW \)) if the reroute request is accepted, namely if the counter in the REROUTE message is larger than the stored counter for this VC, are \textit{VCT\text{-}update} and \textit{VCT\text{-}move}. The \textit{VCT\text{-}update(A_Y)} operation updates entry \( wa_1 \) with \( l_{13} \), the \textit{starting label} of the stand-by VP channel of VP(\( AR \)) and with \( A_L \), the new Link. The \textit{VCT\text{-}move} operation moves entry \( ea_1 \) from VCT(\( A_B \)) to VCT(\( A_L \)). The VCI entries in VCT(\( A_L \)) for the VCs that will use NewVP(\( EA \)) are all empty before this operation because NewVP(\( EA \)) is new at port \( A_L \).
Figure 8: Rerouting of VC(WZ) to a new VP
Switch A sends a REROUTE-ACK message on NewSVC(AE) and then moves VC(WZ) to active state. Upon receiving the REROUTE-ACK message, switch E moves VC(ZW) to active state, i.e., begins to send cells on NewVP(EA). At this time the rerouting procedure for VC(WZ) is complete.

### 3.3 Handling failures during the rerouting procedure

A failure on a link/switch of NewVP is considered as a rerouting-procedure failure if it occurs before the REROUTE message arrives to the place of the failure or if it occurs after the REROUTE message has passed but before the REROUTE-ACK has arrived. A failure that occurs after the REROUTE-ACK has passed is considered as a regular failure.

Rerouting-procedure failures are handled in FRP in the same way as regular failures. The FAIL message of the rerouting protocol informs the endpoint(s) about the failure. The FAIL message will arrive at the rerouting endpoint (switch E in Fig. 8) and will trigger a new rerouting procedure, i.e., search for a new bypass and rerouting over it. This will happen following a rerouting-procedure failure, whether the REROUTE message has or has not previously traversed the location of the failure. Moreover, if the REROUTE has traversed the failure location, the FAIL message will follow the REROUTE message on its travel to the non-rerouting endpoint (switch A in Fig. 8). Switch A will receive the FAIL and will act in the same way as in the regular failure case. In fact, switch A cannot distinguish between a regular failure and a rerouting-procedure failure and does not need to.

Returning to the actions of the rerouting endpoint, for a given VC, the REROUTE messages sent with consecutive rerouting attempts contain ever increasing values of VCcounter. This allows the non-rerouting endpoint to distinguish between REROUTE messages of new and old attempts and to act only on newer ones. The VCcounter takes care of the possibility that an older REROUTE is slow and arrives at the non-rerouting endpoint after a newer one.
3.4 The Rerouting Protocol Pseudo-Code

Protocol Reroute

Messages

FAIL(VPid, location)
REROUTE(NewVPid, NewVP, INFO)
REROUTE-ACK(INFO)

Variables

VPid - global VP-id of the failed VP
NewVP - topology of the new VP
NewVPid - global VP-id of the new VP
INFO - array of VCs; for each VC, the variable VCid is the global VC-id and VCcounter is the value of the rerouting counter for this VC.
location - location of failure
ListOfLocations - a list of failure locations.
timer(VC) - a timer of a given VC

Waiting - a list that contains the VCid’s of the VCs that have not been rerouted yet and for which there is no current rerouting attempt.
Attempting - a list that contains the VCid’s of the VCs that have not been rerouted yet and for which rerouting is being attempted.

Notes:

1. When there is need to distinguish between parameters carried in messages and the same variable stored at a switch S, variables without subscript denote the former, while variables with subscript S denote the latter.

2. VCcounter is set to zero when the VC is established and cancelled when the VC is cancelled.

3. The algorithm is for a given VP and its replacements.
Algorithm for rerouting endpoint $E$ of VPid

A1 Upon receipt of FAIL($VPid$, location)
A2 For each VC that uses $VPid$
A3 If (state = ready) Then activate timer($VC$)
A4 state $\leftarrow$ passive for bidirectional VC
A5 move VC to Waiting
A6 Add location to ListOfLocations
A7 If Waiting is not empty, enter Rerouting mode for this VP

B1 Upon receipt of REROUTE-ACK($INFO$)
B2 For each VC $\in$ INFO
B3 state $\leftarrow$ active for incoming unidirectional VC
B4 deactivate timer($VC$)
B5 remove VC from Attempting
B6 For all incoming unidirectional VC's $\in$ VCTblock - $INFO$
B7 state $\leftarrow$ passive
B8 move VC to Waiting
B9 If Waiting is not empty, enter Rerouting mode for this VP

C1 Upon expiration of timer($VC$)
C2 Cancel VC and remove it from all lists

D1 In Rerouting mode
D2 While Waiting is not empty and NewVP to A that does not traverse any link in ListOfLocations is found
D3 Create VCTblock for NewVP
D4 For each VC $\in$ Waiting
D5 VCT-move on incoming unidirectional VC
D6 VCT-update on outgoing unidirectional VC
D7 move VC from Waiting to Attempting
D8 increment $V$Counter$_E$
D9 include ($VCid_E$, $V$Counter$_E$) in $INFO_E$
D10 Send REROUTE($NewVPid$, NewVP, $INFO_E$) on Signaling-VC to next Main Intermediate Switch of NewVP
D11 For each VC $\in$ $INFO_E$
D12 state $\leftarrow$ active for outgoing unidirectional VC
D13 If (Waiting = Attempting = empty) Then leave Rerouting mode

Algorithm for non-rerouting endpoint $A$ of VPid

E1 Upon receipt of FAIL($VPid$, location)
E2 For each VC that uses $VPid$
E3 If (state = ready) Then activate timer($VC$)
E4 state $\leftarrow$ passive

F1 Upon receipt of REROUTE($NewVPid$, NewVP, $INFO$)
F2 Create VCTblock for NewVP
F3 For each VC $\in$ $INFO$
F4 If ($V$Counter$_A$ < $V$Counter) Then
F5 $V$Counter$_A$ $\leftarrow$ $V$Counter
F6 state $\leftarrow$ passive
F7 VCT-move on incoming unidirectional VC
F8 VCT-update on outgoing unidirectional VC
F9 include $VCid_A$ in $INFO_A$
F10 send REROUTE-ACK($INFO_A$) on Signaling-VC of NewVP
F11 For each VC $\in$ $INFO_A$
F12 state $\leftarrow$ ready

G1 Upon expiration of timer($VC$)
G2 Cancel VC and remove it from all lists
Algorithm for Main Intermediate Switch R of VPId

H1  Upon receipt of REROUTE(NewVP, NewVPid, INFO)
H2  VPT-concat
H3  Send REROUTE(NewVP, NewVPid, INFO) on Signaling-VC to next Main Intermediate Switch of NewVP

4 Main Properties and Protocol Validation

In this section we present and prove the main properties of the rerouting protocol. We use the model and the protocol definitions of previous sections.

Definition 6 A VC cell is forwarded correctly along the VC if it is sent by the first switch in the VC path according to the VC starting label and arrives at the VC ending label in the last switch in the VC path.

Theorem 1 Consider a VC and a VP that is used by this VC, and suppose a finite sequence of rerouting procedure failures occur.

a) VC cells are either discarded or forwarded correctly along the VC.

b) VC cells arrive at their destination in FIFO order.

c) If timer(VC) is long enough, if no failure occurs on the final bypass and system resources exist, then the VC cells are forwarded correctly in finite time.

The Theorem is proved by a series of Lemmas. We shall use the following notation:

- $b_0$ the original bypass that fails
- $b_1, \ldots, b_{n-1}$ subsequent bypasses over which the considered VC is rerouted that fail during the rerouting procedure
- $b_n$ the final bypass that does not fail

Lemma 1 The VPT-concat operations in the main intermediate switches of a bypass construct a VP that is a concatenation of the stand-by VPs.

Proof: Consider VPT-concat($R_H, R_G$) that concatenates the stand-by VP(AR) with VP(RE) as in Figure 6. VPT-concat($R_H, R_G$) replaces the fields of the VP ending label in stand-by VP(AR) with the starting label that contains the VPI value and the port that leads to the first entry of
stand-by VP(RE). All other entries do not change, hence the starting label at switch A and the new collection of entries between switches A and E form a unidirectional VP. Similarly for the other direction. In general, a bypass that is composed of $k$ stand-by VPs has $k - 1$ main intermediate switches. The first VPT-concat operation concatenates the first stand-by VP with the second and thus creates a new stand-by VP. The second VPT-concat operation concatenates the new stand-by VP with the third, and so on.

Lemma 2  Suppose that VPT-concat is performed in the main intermediate switches of some bypass $b_i$, $1 \leq i \leq n$, and VCT-move and VCT-update are performed in the endpoint switches for the given VC for bypass $b_i$. Then the VC is preserved according to the VC definition.

Proof:

For VC(WZ) in Figure 8, switch E performs VCT-move, resulting in a new VCT entry waiting for cells of VC(WZ) on bypass $b_i$. Switch E sends the REROUTE message that activates the VPT-concat operation in the main intermediate switches of bypass $b_i$. This operation constructs a VP as shown in Lemma 1. Upon receipt of the REROUTE message, switch A performs VCT-update for VC(WZ), resulting in an updated VCT entry waiting for cells of VC(WZ) that will arrive at switch A. All other entries do not change, hence the outcome of these operations is a unidirectional VC as defined in Definition 3.

Consider now VC(ZW) at E as in Figure 8. Switch E performs VCT-update, resulting in an updated VCT entry waiting for cells of VC(ZW) that arrive at switch E. Upon receipt of the REROUTE message, switch A performs VCT-move, resulting in a new VCT entry waiting for cells of VC(ZW) on bypass $b_i$. All other entries do not change, hence the outcome is a unidirectional VC.

Definition 7  If a VC cell is transmitted by the VC source after another cell, it is referred to as a newer cell.

Lemma 3  For each direction of VC(WZ), let bypass $b_j$, $1 \leq j \leq n$, be some bypass over which at least one cell is transmitted. Let bypass $b_k$, $0 \leq k \leq j - 1$, be the last bypass before bypass $b_j$ over which at least one cell is transmitted in the same direction. Then, the first cell transmitted on bypass $b_j$ in this direction is newer then the last cell transmitted on bypass $b_k$ in this direction.

Proof: Consider the rerouting of VC(WZ) as shown in Figure 8. The FAIL message that goes to switch E is generated on bypass $b_k$ by the switch near the failure and goes up the ATM management plane at every switch on its way to the end point switch. The last cell transmitted on bypass $b_k$ is sent by the switch near the failure before the FAIL message and is routed over switches only.
Therefore, the FAIL message arrives at switch E after this cell. Moreover, the first cell transmitted on bypass $b_j$ arrives at E after this FAIL, since it can be sent by A only after receiving the REROUTE on $b_j$, which is sent by E after having received the FAIL on $b_k$. This proves the Lemma for the direction from W to Z.

Consider now the rerouting of VC($ZW$). Switch E can transmit cells on $b_j$ only after receiving a REROUTE-ACK message on this bypass, and the latter can be generated by switch A only after having received a REROUTE message on this bypass. Thus the first cell transmitted by E on $b_j$ is sent after the time when A receives the REROUTE message. Upon receiving the REROUTE message on bypass $b_j$, switch A performs VCT-move, that moves the VCT entry to accept cells of VC($ZW$) on bypass $b_j$, and ensures that cells of VC($ZW$) that arrive at switch A on bypass $b_k$, $0 \leq k \leq j - 1$ after this operation, are discarded. Hence, the first cell transmitted on bypass $b_j$ in this direction is newer then the last cell transmitted on bypass $b_k$ in this direction.

**Lemma 4** If $\text{timer}(VC)$ is long enough, if no failure occurs on the final bypass after a finite sequence of failed rerouting procedures and system resources exist, then the VC cells are forwarded correctly in finite time.

**Proof:** Consider the rerouting of VC($WZ$) as shown in Figure 8 and assume that $\text{timer}(VC)$ is long enough and system resources exist. Since no failure occurs on bypass $b_n$, the REROUTE message is sent by switch E on bypass $b_n$ after it performs VCT-move and VCT-update. Since no failure occurs on bypass $b_n$, all the main intermediate switches on bypass $b_n$ perform the VPT-concat operation and forward the REROUTE message. Since $VC\text{counter}_E$ is incremented by E every time a REROUTE message is sent, the value of $VC\text{counter}$ received by A in the REROUTE message on $b_n$ is strictly larger than the current $VC\text{counter}_A$. Therefore, when switch A receives the REROUTE message on $b_n$, it performs VCT-move and VCT-update. This implies, according to Lemma 2, that the VC is preserved and thus the VC cells are forwarded correctly. This procedure is finite.

**Proof of Theorem 1:** If a bypass fails, cells are discarded. If it does not fail, Lemma 2 ensures that the considered VC is preserved and thus cells are forwarded correctly. Thus, part a) holds. Lemma 3 proves that the first cell transmitted on a given bypass in a given direction is newer than the last cell transmitted in the same direction on an earlier bypass. This ensures FIFO order of the cells over the VC, hence b). Lemma 4 proves that if $\text{timer}(VC)$ is long enough, if no failure occurs on the final bypass after a finite sequence of failed rerouting procedures and system resources exist, then the VC cells are forwarded correctly in finite time. Thus, part c) holds.
5 Numerical Results

In this section we compare the rerouting times for three schemes: FRP, the backup VP scheme and complete VP reconstruction. In addition we give an estimate of the probability of finding an alternate route of concatenated stand-by VP's in a network.

For the comparison, we assume a bypass for a given VP that is built of 5 stand-by VP's, each consisting of 4 links on the average and whose total geographic length is 1000 km. The rerouting time, in msec, for each method is estimated as in Table 1:

<table>
<thead>
<tr>
<th></th>
<th>backup VP</th>
<th>FRP</th>
<th>VP reconstruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>bypass search</td>
<td>0.2</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>propagation time</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>VCT operations</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>concat operations</td>
<td>1.6</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>switch time for intermediate</td>
<td>1</td>
<td>2</td>
<td>9.5</td>
</tr>
<tr>
<td>switch time for endpoints</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>total</td>
<td>10.6</td>
<td>14.4</td>
<td>27.9</td>
</tr>
</tbody>
</table>

Table 1: The rerouting time, in msec, for the different methods

The bypass search time is the time it takes the rerouting endpoint to find a path. We estimated this time by writing a simulation C++ program on a SUN Ultra4 for finding a bypass in the UUNET High-Speed backbone \(^1\) with random VP layout. We have measured this time as 0.2 msec. The propagation time of the REROUTE and REROUTE-ACK messages depends on the geographic length of the bypass and in our example is 4 msec in each direction \([8]\). We also implemented the VCT and VPT operations and measured the processing time of the VCT-move and VCT-update in one endpoint switch as 0.8 ms and the VPT-concat operation in each intermediate switch as 0.4 ms. Therefore, the concatenation operations take \(4 \times 0.4 = 1.6\) msec in FRP and \(19 \times 0.4 = 7.6\) msec in the VP reconstruction scheme. The intermediate nodes need also to handle the REROUTE and REROUTE-ACK messages, that takes about 0.5 msec \([3]\). Therefore the total time is \(4 \times 0.5\) and \(19 \times 0.5\) msec respectively. The total rerouting time is therefore 14.4 msec in FRP compared to 10.6 in the backup VP protocol and 27.9 msec for total VP reconstruction.

In addition to the time estimates, we are interested in estimating how often can one find a bypass consisting of a concatenation of stand-by VP's. For this purpose we have again used the UUNET High-Speed backbone topology that consists of 33 switches and 128 links. The program randomly selects a number \(X\) of VP's in the network and a percentage \(p\) of unused VP's (standby VP's). Then we find the number of cases when a concatenation of standby VP's can be found to bypass a single switch failure. The results for \(X = 100, 200, 300, 400\) and 500 and for \(p = .35, .5\) and .75 are given in Fig. 9.

\(^1\)Topology map is available from http://www.uunet.net/lang.en/network/us.shtml
6 Conclusion

We have proposed a protocol for flexible rerouting of VCs when the underlying VPs fail in ATM networks. The protocol is a trade-off between the backup VP protocol, that requires stringent management of backup VP’s and the complete VP reconstruction protocol, that has a large latency. The protocol uses simple operations at the endpoints of the rerouted VP and of the backup VP’s to perform the construction of the new VP and the rerouting of the VC’s. Calculations show that the extra rerouting time of FRP compared to the backup VP is small, while providing greater management flexibility.

References


