Path Merging and Destination Removal:
New Forwarding Paradigms in Packet Switching Networks
Shai Benjamin and Izhak Rubin
University of California, Los Angeles

Abstract -- We study how two new forwarding paradigms, namely path merging and destination removal, help in reducing the number of labels in the routing table of a packet switching node. We prove upper and lower bounds on the number of labels required for general networks and give tractable solutions for some specific cases (tree, cycle, complete graph). We believe that destination removal, as an extension to Multiprotocol Label Switching (MPLS), may result in further enhancement of the performance of high-speed networks.

Index terms-- routing, virtual path layout, ATM, MPLS, graph theory, NP-complete.

I. INTRODUCTION

As the bandwidth of physical links increases dramatically, the processing speed of nodes becomes a bottleneck in the performance of telecommunication networks. One of the most demanding tasks of a packet switching node is, of course, the fast routing and switching of packets. Next hop decisions are made based on a lookup table indexed by destination addresses or path labels. The size of the address space or label space determines the size of the routing table, as well as the time it takes to perform search operations on these tables. Reducing the size of the routing table is therefore a major factor in accelerating the operations of a packet switching node.

A virtual path layout (VPL) is a collection of virtual paths (VPs) designed such that each pair of nodes in the network can be connected by a virtual circuit (VC) which is a concatenation of no more then a prescribed number of VPs. Using a VPL reduces the size of routing tables in the nodes, resulting in faster and more efficient routing operations, as mentioned above.

VPLs are usually designed under some combination of requirements and objectives, including:
1. Topological constraints: the VPL is required to provide connectivity between each pair of nodes with a given maximum number of VP hops.
2. Performance: the size of the VP routing tables at each node should be minimized (alternatively, the total number of VPs in the network should be minimized).
3. Capacity: the VPL needs to accommodate some given traffic matrix, by assigning some capacity to each VP.
4. Redundancy: the VPL is required to be k-connected, i.e., allow k disjoint routing alternatives for each pair of nodes in order to satisfy some reliability requirements as well as to allow traffic balancing.

The problem of designing a VPL for ATM networks is a well known optimization problem, treated extensively in the communication and graph theory literature. Numerous works were published suggesting various heuristic approaches for solving the VPL design problem ([1-8] and references therein, to mention just a few). In all these works, a VP is a simple path, i.e., an end-to-end directed route between two nodes of the network. A packet cannot use a portion of a VP. It must traverse a VP from its designated starting node to its terminating node, at which it is either switched (hopped) to another VP or removed from the network (if it reached its egress node).

In this paper we introduce structures other than simple paths to be used as vehicles for forwarding packets in a packet switching network. We call these Unambiguous Directed Subgraphs (UDs) and note that a VP is a special case of a UD. Similarly, we use UDL instead of VPL.

The inception of Multiprotocol Label Switching (MPLS) [10-13] introduced the new concept of path merging (or label merging) to the UDL design problem. With this forwarding paradigm, a UD need not be a simple path but can admit the structure of a directed in-tree. For single UD-hop routing (i.e., providing connectivity between each pair of nodes in the network without having to switch between UDs along the route), this may reduce the global number of labels required to identify the UDs from \(O(n^2)\) in the single VP-hop VPL design problem to \(O(n)\).

We introduce here a second new concept called destination removal. With this forwarding paradigm, a valid UD may also admit the structure of a functional digraph [9] (see definition below). We will show how the combination of path merging and destination removal may significantly reduce the number of required labels in the routing tables of the network nodes.

In section II of this paper we describe our model and explain the details of the new forwarding paradigms. In section III we define the UDL design problem and in section IV we review its solutions for the different forwarding paradigms. In section V we concentrate on the case when both path merging and destination removal are used and in section VI we give some tractable examples for this case. Section VII concludes this paper.
II. NEW FORWARDING PARADIGMS

We model the underlying communication network as an undirected graph \( G = (V, E) \) where \( V \) denotes the set of nodes and \( E \) denotes the set of links or edges. We will use the terms network and graph interchangeably. Let \( n = |V| \) denote the number of nodes in \( V \). A UD is a directed subgraph of \( G \) for which the outgoing degree of each node is at most one. A UD is not necessarily connected. This allows spatial reuse of UD identifiers (labels).

We define the following forwarding paradigms:

**Path Merging**

When path merging (PM) is allowed, each UD in the UDL is a directed tree where the direction of flow is towards the root of the tree (an in-tree). The leaves of the tree are the starting nodes of the UD and its root is the ending node. A UD can have more than one starting node but only one ending node. Packets can enter the UD at any node. Each UD is identified by a label. At each node, the routing table is keyed by the labels of those UDs that traverse the node and it includes one entry for each key which identifies the next-hop node for the UD, or a marker that identifies the node as terminating that UD. When sending a packet from node to node, the source node labels the packet with an identifier that uniquely corresponds to the UD which terminates at the destination node. When the packet arrives at an intermediate node along the UD it is forwarded towards its destination according to the routing table. Only the UD's designated destination node can terminate the packet. In the current ATM protocol, as well as tag-switching or label-switching protocols, no modification is required to allow path merging. PM is implemented by simply having the same next-hop identifier keyed more than once in the routing table.

We will use PM when path merging is not allow.

**Destination Removal**

When destination removal (DR) is used, a packet can enter or leave a UD at any node along the path and not necessarily at the starting or ending nodes. Each UD is either a simple path or a cycle in the underlying network. At each node, the routing table is keyed by the labels of those UDs that traverse the node and it includes two entries for each key. The first entry is a node sublabel and the second entry identifies the next-hop node for the UD. When sending a packet from node to node, the source node labels the packet with two identifiers. One identifier corresponds to a UD which traverses the destination node (but does not necessarily terminate there). The other identifier is a destination sublabel which identifies the destination node on that UD. When the packet arrives at an intermediate node along the UD, its destination sublabel is compared with the node sublabel. If the two sublabels match then the packet is terminated. If not, the packet is forwarded towards its destination according to the second entry in the routing table. Note that destination sublabels have a context only with regard to their UDs and therefore they can be reused on different UDs. Later we will show that sublabels within the same UD need not be unique. An alternative way for implementing destination removal is using a time-to-live (TTL) field. Every time the packet is forwarded by a node, the value of the TTL field is decrement by 1. The packet has reached its destination when the TTL field equals zero, and it can then be removed.

We note that destination removal requires a simple comparison operation for each routed packet. However, as we will show, destination removal may significantly reduce the size of the routing table, thus making the search for entries in the table much faster and offsetting this inconvenience.

We will use DR when destination removal is not allowed.

**Path Merging & Destination Removal**

Allowing both path merging and destination removal (denoted PMDR) introduces an even more powerful tool when designing UDLs. UDs in this case admit the structure of an in-tree or a functional digraph (a digraph with outdegree at each node exactly one, see example in Figure 1). Packet forwarding is the same as in the DR case, using UD labels and destination sublabels. Note that a functional digraph has exactly \( n \) edges and it includes exactly one cycle in each one of its (connected) components. We define the girth of a functional digraph as the number of edges of its minimal cycle.

We look at 4 types of networks using the following notations:

- **PMDR**: The network doesn't allow path merging or destination removal. ATM is an example of this kind of network.
- **PMG**: The network allows path merging but doesn't allow destination removal. MPLS is an example of such a network.
- **PMDR**: The network allows destination removal but doesn't allow path merging. One example when path merging might not be allowed is an all optical network, which cannot tolerate queuing [14, 15].
- **PMDR**: The network allows both path merging and destination removal. PMDR is applicable for MPLS networks with the appropriate modifications to accommodate destination removal (namely, adding the field and handling of destination sublabels or TTL).
III. THE UDL DESIGN PROBLEM

Requirements
- We focus on objectives 1 and 2 of section I, namely, the design of a UDL which provides connectivity between all pairs of nodes in the network while minimizing the total number of UDs in the network.
- We require that the path induced by the UDL between each pair of nodes in the network is a shortest path between that pair (i.e., a stretch factor of 1 [3]).
- We assume routing decisions at the nodes ignore the port identifier on which a packet arrives. Only the UD total number of UDs in the network.
- We assume that the path induced by the UDL between each pair of nodes in the network is a shortest path between that pair (i.e., a stretch factor of 1 [3]).
- We assume routing decisions at the nodes ignore the port identifier on which a packet arrives. Only the UD total number of UDs in the network.

The UDL Design Problem
Consider an undirected graph \( G = (V,E) \), integers \( h,k \geq 1 \). We seek a UDL with no more than \( k \) UDs which provides shortest-path connectivity between each pair in \( V \) with no more than \( h \) UD-hops.

In this paper we deal with the single UD-hop UDL problem \((h = 1)\), i.e., UD-hopping is not allowed so that routing between each pair of nodes must be possible using exactly one UD in the UDL.

We denote by \( k^*(G) \) the optimal (minimal) \( k \) for which a solution to this problem exists. We are interested in the value of \( k^*(G) \) as well as a realization of a UDL that achieves \( k^*(G) \). Note that an optimal UDL is not necessarily unique.

IV. SOLUTIONS TO THE UDL DESIGN PROBLEM

In this section we look at the 4 types of networks defined in section II and examine the solution to the UDL design problem for these networks.

1. \( \text{PMDR} \)
The UDL design problem for \( \text{PMDR} \) is actually the same as the conventional VPL design problem. We note that when destination removal is not allowed a VP can only serve one node as a destination node. Therefore, since each node must be able to forward packets to \( n-1 \) other nodes, and also be able to terminate packets for which it is the egress node, \( n \) is a lower bound on the size of the routing table at each node. A path is required from each node to all other nodes and therefore there will be exactly \( n(n-1) \) virtual paths in the VPL. Since some of these VPs are spatially (node) disjoint, labels can be reused on groups of disjoint VPs to form UDs. Still, the total number of UDs in the UDL is \( k^*(G) = O(n^2) \) [3].

2. \( \text{PM} \bar{D} \bar{R} \)
The single UD-hop \( \text{PM} \bar{D} \bar{R} \) problem turns out to be a very simple one. For each node \( v_i \), we construct the in-tree of shortest paths (we will use TrSP for tree of shortest paths) from all nodes of the network to \( v_i \) and denote it \( T_i \). Then the collection of trees \( \{ T_i \}_{i=1}^{n} \) is the set of UDs in the UDL. The size of the routing table at each node is exactly \( n \), and since this achieves the lower bound we stated in 1 above, this is the optimal solution.

3. \( \text{PMDR} \)
Destination removal with no path merging allows only cycles and simple paths as UDs. For example, only two UDs are required in a ring network, one that carries packets in the clockwise direction and the other in the counter-clockwise direction. We leave the \( \text{PMDR} \) problem outside the scope of this paper and only mention that in some cases an optimal \( \text{PMDR} \) UDL exists which is also an optimal \( \text{PMDR} \) UDL.

4. \( \text{PMD} \)
As we mentioned before, allowing both path merging and destination removal provides a very powerful tool for designing UDLs. We will explore the UDL design problem for \( \text{PMD} \) in the following section.

V. PATH MERGING AND DESTINATION REMOVAL

In this section we give upper and lower bounds for \( k^*(G) \) in the \( \text{PMDR} \) case. We will show a general method for solving the \( \text{PMDR} \) UDL design problem and then give some examples of solutions.

Lower Bound
Consider a node \( v \in V \). Since the outdegree of each node in a UD is at most one, routing from \( v \) to each one of its neighbors must be done over a distinct UD, otherwise routing is not over the shortest path. This leads to the following result:

Proposition 1: A lower bound for \( k^*(G) \) is
\[
\Delta(G) = \max_{v \in V} \deg(v).
\]

A tighter lower bound can be constructed as follows: Let \( l(T) \) denote the number of leaves of a tree \( T \) and \( \mathcal{T}(v) \) the collection of all out-trees of shortest paths originating from a node \( v \) to all other nodes in \( V \). Consider a node \( v_i \in V \) and pick a tree \( T_i \) from \( \mathcal{T}(v_i) \) that satisfies \( l(T_i) = \min_{T \in \mathcal{T}(v_i)} l(T) \). At least \( l(T_i) \) labels are required to uniquely identify the routes from \( v_i \) to the leaves of \( T_i \). Using destination removal, the same paths can be used to reach all nodes which are not leaves. This leads to the following result:

Proposition 2: A lower bound for \( k^*(G) \) is
\[
\max_{v \in V} \min_{T \in \mathcal{T}(v)} l(T).
\]

The problem of finding a TrSP with minimum number of leaves can be solved in polynomial time.
Upper Bound

Since \( \text{PMDR} \) is a special case of \( \text{PMDR} \), \( n \) is an upper bound for \( k^*(G) \). We will give a simple construction of a UDL with only \( n-1 \) UDs and use this construction for a possibly smaller upper bound.

**Proposition 3:** For every network \( G = (V, E) \), a UDL with \( k = n-1 \) exists.

**Proof:** Pick a node \( v \in V \). Construct the \( n-1 \) in-TrSPs for all nodes but \( v \). Choose one of \( v \)'s neighbors, say \( v_i \), and add an edge to its in-tree which extends from \( v_i \) to \( v \). The resulted digraph has out-degree exactly 1 for all nodes (it is a functional digraph) and therefore is a valid UD. Repeat this for all neighbors of \( v \). Since \( v \) is reachable from every node of the network over a shortest path embedded in at least one of the constructed functional digraphs, the set of \( n-1 \) UDs is a UDL.

For a better bound we define the **dominated set** of node \( v \) as the node \( v \) itself and all its neighbors, denoted \( \text{dom}(v) \). We call \( v \) the **set leader**. Let \( n_d(G) \) denote the maximum number of disjoint dominated sets that can be found in a graph \( G \) and define the collection of set leaders that achieve the maximum as the **maximal dominating set**.

**Proposition 4:** An upper bound for \( k^*(G) \) is \( n - n_d(G) \).

**Proof:** Find a maximal dominating set for \( G \). Construct in-trees of shortest paths for each node of \( G \) not in the maximal dominating set (there are \( n - n_d(G) \) such nodes). Repeat the construction of proposition 3 for each set leader and its neighbors. The resulting set of UDs is a UDL.

Note that the problem of finding \( n_d(G) \) is the known set packing problem (which is NP-Complete) where the subsets are all sets of the form \( \text{dom}(v) \).

**VI. EXAMPLES**

In this section we describe some cases for which a tractable solution to the **PMDR UDL** design problem exists. Those include trees, complete graphs and cycles. We also describe cases that can be decomposed or reduced to smaller **PMDR** problems.

**Trees**

We use the lower bound found in proposition 2 and show that the optimal UDL for a tree \( T \) has \( l(T) \) UDs.

**Proposition 5:** For a tree \( T \) with more than 2 nodes, \( k^*(T) = l(T) \).

**Proof:** Let \( T = (V,E) \) be a tree with \( |V| > 2 \). For any node \( v \in V \), a TrSP from \( v \) to all other nodes in \( T \) is a proper orientation of \( T \) with \( l(T)-1 \) leaves if \( v \) is a leaf, and \( l(T) \) leaves if \( v \) is not a leaf. Since there is at least one node in \( V \) which is not a leaf, a lower bound for \( k^*(T) \) is \( l(T) \). A **PMDR UDL** that achieves this bound is the set of \( l(T) \) in-TrSPs whose destination nodes are the leaves of \( T \). To prove this is indeed a UDL we need to show a shortest path is embedded in at least one member of the UDL for any pair \( s, d \in V \) of source and destination nodes respectively. If \( d \) is a leaf, then since a path between two nodes of a tree is unique, the path from \( s \) to \( d \) is included in the TrSP from all nodes to \( d \). If \( d \) is not a leaf, then construct the unique directed path from \( s \) to \( d \). Since \( s \) is not a leaf, i.e., \( \deg(d) > 1 \), we can extend this path by adding to it a directed edge from \( d \) to one of its neighbors which is not already in the path. If the node we reached is a leaf then by the same argument the proof is complete. Otherwise we continue the construction until we eventually reach a leaf.

Note that a line graph \( L_n, n > 2 \) (also called a "snake") is a special case of a tree and the UDL contains only two UDs, one for each end-node.

**Cycles**

For a cycle \( C_n \), we have \( k^*(C_n) = 2 \), regardless of \( n \). The two UDs in the optimal UDL are the embedded clockwise directed cycle and the counter-clockwise cycle. The fact that this is a UDL is obvious. It is the optimal UDL as it achieves the lower bound of proposition 1.

On the other hand, \( C_n \) demonstrates the possible looseness of the upper bound given in proposition 4, which equals \( \left\lceil \frac{2}{3}n \right\rceil \) in this case.

**Complete graph**

For a complete graph \( K_n \), both lower and upper bounds of propositions 1 and 3 (respectively) equal \( n-1 \).

One UDL with \( n-1 \) UDs can be constructed as described in the proof of proposition 3. A more elegant UDL is constructed as follows:

Denote \( V = \{v_i\}_{i=0}^{n-1} \). Repeat the following for \( j = 1 \rightarrow n-1 \):

- Construct the directed cycle \( v_0 \rightarrow v_j \rightarrow v_{2j} \rightarrow \ldots \rightarrow v_0 \) (indices are modulo \( n \)). If this cycle does not cover all the nodes in \( V \), then also construct \( v_{1} \rightarrow v_{3+j} \rightarrow v_{5+2j} \rightarrow \ldots \rightarrow v_{1} \) and so on, until the set of cycles covers \( V \). For each \( j \), the collection of cycles is a functional digraph. Call it
Proof of Proposition 5

The proof of proposition 5, equals $K_2$, and therefore has one UD in its optimal UDL.

1-point-connected graphs

Assume the graph $G = (V, E)$ contains a cut-point $v$ that when removed from $G$ results in two or more disconnected components $G_1 = (V_1, E_1), \ldots, G_n = (V_n, E_n)$ where $\{v\} \subseteq V_1 \ldots \subseteq V_n$ and their union is $V$. Denote by $G_a = (V_a, E_a)$ the subgraph spanned by $\{v\} \cup V_1$ and denote by $G_b = (V_b, E_b)$ the subgraph spanned by $\{v\} \cup (\bigcup_{i=2}^n V_i)$. To solve the UDL problem for $G$, we first solve the two smaller problems of designing UDLs $U_a = \{U_i^a\}_{i=1}^m$ and $U_b = \{U_i^b\}_{i=1}^m$ for $G_a$ and $G_b$ respectively. We then combine those into one UDL for $G$ as follows:

Let $T_a$ be the TrSP from all nodes in $V_a$ to the cut-point $v$ and $T_b$ the TrSP from $V_b$ to $v$. One of the UDs in $U_a$ is a TrSP from all nodes in $V_a$ to $v$ then remove this UD from $U_a$. Similarly for $U_b$ with respect to $V_b$. A UDL for $G$ is then $U = (U_a + T_a) \cup (U_b + T_b)$ where $A + B$, for $A, B$ subgraphs of $G$, is the subgraph that includes all nodes and edges of $A$ and $B$, and $A + B$, for $A$ a collection of subgraphs and $B$ a subgraph, is the collection of all subgraphs of the form $A + B$ for all $A \in A$. Note that if $A$ is a functional digraph (tree) and $B$ is a tree and they have exactly one common node, then $A + B$ is also a functional digraph (tree).

Proposition 6: $U$ is a UDL for $G$.

Proof: For each pair of nodes which are both in $G_a$, there is a UD that connects the two points over a shortest path by construction of $U_a$, unless the UD for these two points was the one removed from $U_a$ in the construction of $U$.

For this case, any UD of the form $U_i^b + T_a$ provides shortest path connectivity between the pair over the edges of $T_a$. Similarly for $G_b$, $T_b$ and $U_b$. Now assume $s \in G_a$ and $d \in G_b$. There is a UD $U_i^b$ that connects $v$ to $d$ over a shortest path and $T_a$ connects $s$ to $v$ over a shortest path and finally, since $v$ is a cut-point, there is a path which is a shortest path from $s$ to $d$ in $U_i^b + T_a$ which is a member of $U$.

We note, without a proof, that if $U_a$ and $U_b$ are optimal, then so is $U$.

VII. CONCLUSIONS

In this paper we present the UDL design problem for packet switching networks that incorporate the forwarding paradigms of path merging and destination removal. We concentrate on the single UD-hop problem. For a network with path merging and no destination removal (PM DR) we give a complete solution which is the $n$ in-trees of shortest paths, one tree for each node. We give upper and lower bounds for the case when both path merging and destination removal are used (PMDR) and complete solutions for some specific cases. We believe that destination removal, as an extension to MPLS, may result in further enhancement of the performance of high-speed networks.

We intend to extend our work and address fault tolerant (i.e. $k$-connected) networks, hierarchical routing (UD-hop count greater than 1) and traffic engineering. We also plan to look at general topology PDR (e.g. all optical) networks.

REFERENCES

9. Frank Harary "Graph Theory" Addison Wesley 1969
11. IETF, "A Framework for Multiprotocol Label Switching,