# Prefix routing schemes in dynamic networks

E.M. Bakker, J. van Leeuwen, R.B. Tan

RUU-CS-90-10
March 1990


Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

ISSN:0924-3275

# Prefix Routing Schemes in Dynamic Networks *

Erwin M. Bakker , Jan van Leeuwen
*Department of Computer Science, University of Utrecht*
*P.O.Box 80.089, 3508 TB Utrecht, the Netherlands*

Richard B. Tan
*Department of Computer Science, University of Utrecht*
*and*
*University of Science & Arts of Oklahoma*
*Chickasha, OK 73018, USA*

**Abstract**

We study a routing scheme on dynamic networks called Prefix Routing scheme. The scheme is an abstraction of source routing. It assigns fixed addresses to the nodes and one address per link. Routing of messages is done by sending the messages out via the link with maximum common prefix with the destination node. Arbitrary insertions of links and nodes are feasible with constant adaptation cost. It is shown that any dynamic growing network can be assigned such a scheme. We characterize completely the type of fixed networks with dynamic links (i.e. the cost of the links can vary over time) and dynamic networks with arbitrary insertion and deletion of nodes and links (without disconnecting the network) that allow optimum routing in this scheme. A hierarchical routing scheme where each link may carry more than one address is also introduced and the connections between Prefix Routing and static Interval Routing are presented.

## 1 Introduction

In a network such as UUCP, *source* or *path routing* is used [NL83, QH86, MM88]. This type of routing assumes that each message explicitly or implicitly contains an address that specifies a particular path that it wants to follow, for example, *ruuinf!hp4nl!mcsun!uunet!samsung!think!yale!.....* Thus the address of a node in a

*This work was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM).

message is a sequence of names consisting of strings of characters with suitable separators. A routing table is kept at each node, consisting of $O(n)$ or fewer addresses, where $n$ is the number of nodes in the network. The next name in the sequence, i.e. a prefix of the address, is then extracted and the routing table is consulted for this address. It is a relatively simple scheme, easy to implement and allows for easy updates upon expansion of the network.

We study an abstract version of this routing scheme and call it the *Prefix Routing scheme.* Basically, each node is labeled with a unique address consisting of a string of symbols. As a node receives a message that needs to be routed, it checks for a prefix of the destination address and consults its routing table for this prefix address. It then routes the message accordingly to the next node. We would like to keep the local routing table relatively small, say of the order of the degree of the node. Furthermore we would like the scheme to be *dynamic.*

In a dynamic environment, networks evolve with topological changes like the insertion and deletion of nodes and links. The cost of sending messages via a link may also vary over time, so even with no change in the topology of the networks, the routing tables may need to be updated. The known schemes for changing the routing tables per topological or link cost change based on a broadcasting of updates to the whole network, involve considerable adaptation time and message costs. Even the simple case of just inserting a node requires $O(D)$ time and $O(e)$ messages, where $D$ is the diameter and $e$ is the number of links of the network. This explains the current interest for simpler and/or faster algorithms, based on a suitable combination of the naming and routing regimes.

It turns out that Prefix Routing is a natural dynamic scheme. Insertion of nodes and links can be done quite easily. We present various algorithms that assign fixed labels to the nodes and links of any network (graph) such that a Prefix Routing scheme is possible. They all have an optimum adaptation cost of $O(1)$. Not all networks can have a Prefix Routing scheme with optimum routes, no matter how we label the nodes and the links. We show that only networks with biconnected components of size 3 or less can have an optimum Prefix Routing scheme, in the case that arbitrary insertions and deletions of nodes and links (without disconnecting the network) are allowed. If the networks are fixed, so that there are no insertions or deletions of nodes and links, then the networks that have an optimum Prefix Routing scheme are precisely those with biconnected components of size 4 or less. There are also limitations on the size of the address for any Prefix Routing scheme. For some networks, the address size of some nodes can be as high as $O(D \log \Delta)$, where $\Delta$ is the maximum degree among the nodes of the network.

There is a natural connection between the Prefix Routing scheme and the "compact" *Interval Routing scheme.* Interval Routing schemes have been studied and applied in [SK85, vLT86, vLT87, FJ88, FJ86, PU88, ABLP89, AGR89], in the design of compact routing tables. In an Interval Routing scheme, a suitable address is selected from the interval $[1..n]$ for each node and each link is assigned a label which represents a unique interval from $[1..n]$ (viewed as a cyclic name space). The routing

table at each node consists of a table of the address intervals of the links, thus of $O(\Delta)$ addresses. At each node a message with destination $s$ will be routed via the link that contains $s$ in its interval. It is shown in [vLT86] that every fixed graph can be assigned a cycle-free, hence valid Interval Routing scheme, though not necessarily with optimum routes. Optimum routing schemes have been designed with unit cost links for e.g. fixed trees, rings, grids and complete bipartite graphs [SK85, vLT87]. For fixed networks with arbitrary link costs, outerplanar graphs and c-composable graphs also have optimum Interval Routing schemes [FJ86, FJ88]. Furthermore, the Interval Routing scheme has been adapted for dynamically growing trees [AAPS87, AGR89]. In [AGR89] a dynamic routing scheme is given for trees with $O(\log n)$ addresses per link with address size of $O(\log^2 n)$ bits. Its adaptation cost is $O(\log n)$, amortized over the number of changes in the network.

The rest of the paper is organized as follows. In Section 2 we describe the Prefix Routing scheme and show that any network can be assigned a Prefix Routing scheme that guarantees message-delivery. We show how arbitrary insertions of nodes and links can be performed on the network with optimum adaptation cost. In Section 3 we give a complete characterization of the types of fixed and dynamic networks with dynamic links that have optimum Prefix Routing schemes. In Section 4 we introduce hierarchical Prefix Routing schemes, where each link may have more than one address. We also give the connection between the Prefix Routing scheme and the Interval Routing scheme. Finally, we state some open problems and suggestions for future research in Section 5.

# 2 Prefix Routing Scheme

## 2.1 Preliminaries

The model we use is the point-to-point communication model. The network is connected and asynchronous. Each node (site) can only communicate with its direct neighbors via the bidirectional links. We assume that links and nodes do not fail. When costs are considered, the cost of every link is non-negative and *dynamic*, i.e., costs can vary over time. Moreover, links and nodes can be inserted and deleted arbitrarily at any point in the network. In this section we only consider *growing* or *semi-dynamic* networks, i.e., only insertions of nodes and links are allowed. In general, deletions of links or nodes are more part of the study of faulty networks which can destroy the validity of the routing scheme and may require some restructuring of the scheme. However in the next section on optimum routing, we do consider arbitrary deletions of nodes and links for special types of networks.

In this section we develop a number of basic schemes and discuss some of their properties. Let $\Sigma$ be a set of symbols and $\Sigma^*$ the corresponding set of strings over the alphabet $\Sigma$ . We assume throughout that $\Sigma$ contains at least two symbols.

**Definition 2.1** *A $\Sigma^*$-labeling scheme for a network $G$ is a scheme for labeling all*

*the nodes and links in G such that (a) all nodes get unique labels from $\Sigma^*$ and (b) at every node the links get distinct labels from $\Sigma^*$.*

Thus the Interval Labeling scheme is just a $\Sigma^*$-labeling scheme, where $\Sigma = \{1, 2, ...\}$, satisfying the *Interval Property*: at every node and for every destination address there is a link with label $x$ such that the interval specified by $x$ contains the destination address. Wraparound of intervals is allowed. The message with the specified destination address is then routed over that special link to the next node. The corresponding *Prefix Property* is: at every node $u$ of the network $G$ and for any node $v$ there is a link label $x$ at $u$ that is a prefix of the address label $\alpha(v)$ of $v$. Any $\Sigma^*$-labeling scheme that satisfies the Prefix Property is called a $\Sigma^*$-*Prefix Labeling Scheme.*

The basic idea of the Prefix Labeling scheme is to assign a label consisting of a string, over some alphabet $\Sigma$ , to each node. For each link of a node, a distinct label $l_i$ consisting of a string of symbols is attached. All these informations are stored in a local *routing table.* When a message arrives at a node $u$ with destination $v$ (distinct from $u$), the routing table of labels $l_1, ..., l_d$ is searched for the maximum string $l_i$ such that $l_i$ is a prefix of $\alpha(v)$. For example, if $\alpha(v) = 0110$ and $l_1 = \epsilon$ (the empty string), $l_2 = 0$ and $l_3 = 0111$ then $l_2$ is the label that matches the requirement. The message is then routed via the link labeled by $l_2$.

Formally, we have the following recursive routine that models the routing at each node.

**Procedure** *SEND(source,dest,m)*
{ $m$ a message to be sent from the node labeled *source* to the node labeled *dest* }
    **if** *source* = *dest*
    **then** process $m$
    **else**
        find maximum link label $x$ such that $x$ is a prefix of *dest* ;
        send $m$ over the link labeled $x$ ;
        *source* := label of node arrived at over link $x$ ;
        *SEND(source,dest,m)*
    **endif**
**end.** { *SEND* }

**Definition 2.2** *A $\Sigma^*$-Prefix Labeling scheme for a network G that, for any v, routes a message from any node* u *to* v *in G correctly by using the link with maximum label that is a prefix of the address of* v, *(i.e. as prescribed by the procedure SEND) is called a* valid $\Sigma^*$-Prefix Labeling scheme *or simply a* $\Sigma^*$- Prefix Routing scheme.

We often drop the $\Sigma^*$ if no confusion can arise.

## 2.2 Tree Labeling Scheme

There is a simple way of labeling a dynamically growing network using a spanning tree. We call this a *Tree Labeling scheme.* Assume that $\Sigma$ has at least two symbols, say *0* and *1*. Start at an arbitrary node $u$ and label it with *1*. A neighboring node $v$ that is to be attached to $u$ will be named with address *101*. The link from $u$ to $v$ is labeled *101* and that from $v$ to $u$ is labeled $\epsilon$. At the same time node $u$ keeps a count of the number of nodes that requested insertion so far. The next node to be attached to $u$ will be named with address *1001* and the corresponding links will be labeled with *1001* and $\epsilon$ respectively. In the meantime the counter of insertions is again incremented by one. Thus the succeeding neighbors of $u$ will have as addresses *101*, *1001*, *10001*, *100001*, ... i.e., the address of $u$, followed by a list of *0*'s specified by the counter at the moment of insertion, and a *1*. Now apply this process recursively to all the neighbors of $u$. In general, if $\alpha(v)$ is the address of a node $v$ and $w$ is an unlabeled node that needs to be attached to $v$, then $w$ will get an address consisting of $\alpha(v)$, followed by a number of *0*'s specified by the counter at $v$, and a *1*. The corresponding link labels will be $\alpha(w)$ and $\epsilon$, respectively. However, if $w$ is a labeled node and a link is created that attaches it to a previously labeled node $v$ , then the link $(v, w)$ is labeled with $\alpha(w)$ and link $(w, v)$ with $\alpha(v)$. We call such a link a *bypass* link. The above procedure can be easily made precise.

**Initialization:**
    Pick a node $u$ and label it *1* ;
    $c(u) := 1$
**end;**

**Procedure** *INSERT(v,u)*
{ Insert a link from node $v$ to node $u$ . Node $u$ has address $\alpha(u)$
and a counter value of $c(u)$ initialized at 1. *l(v,u)* is the label for link *(v,u)* }
    **if** $v$ is a new node
    **then**
        $\alpha(v) := \alpha(u)0^{c(u)}1$ ;
        *l(v,u)* $:= \epsilon$ ;
        *l(u,v)* $:= \alpha(v)$ ;
        $c(v) := 1$ ;
        $c(u) := c(u) + 1$ ;
    **else** { bypass link }
        *l(v,u)* $:= \alpha(u)$ ;
        *l(u,v)* $:= \alpha(v)$
    **endif**
**end.** { *INSERT* }

The validity of this Labeling scheme for routing is expressed in the following result.

**Theorem 2.3** *There is a valid Prefix Labeling scheme for any dynamically growing network.*

**Proof.** Observe that in the labeling procedure above, every node has an $\epsilon$-labeled link, except the root node *1*. In fact the $\epsilon$-labeled links give a spanning tree of the network with root *1*. Suppose node $u$ needs to send a message to node $v$. There are 3 cases:

1. *$u$ is an ancestor of $v$* : Then there must be a link incident to $u$ with a unique non-$\epsilon$ label that is a prefix of $\alpha(v)$ in the spanning tree or, a bypass link to a descendant that is an ancestor of $v$ or to $v$ directly labeled with a longer prefix of $\alpha(v)$. This condition is true for every node that is on the path from $u$ to $v$ in the spanning tree. Thus a message to $v$ will be routed correctly.

2. *$u$ is a descendant of $v$* : Then none of the links in the spanning tree at $u$ will have a label that is a prefix of $\alpha(v)$ except the $\epsilon$-link to the ancestor. Thus the message will follow the $\epsilon$-links up the tree in the absence of a bypass link. If there is a bypass link up to $v$ directly or to some ancestor of $v$ higher up the tree, the message will be routed either to $v$ directly or to its closest ancestor up the tree due to the maximum prefix condition. From there we get Case 1 again.

3. *Neither of the above cases* : Then $v$ belongs to a different subtree in the spanning tree, so the message will be routed up the spanning tree as in Case 2 until an ancestor or a bypass link to $v$ or to an ancestor of $v$ is encountered. Finally the path as in Case 1 will be followed.

□

The above routing scheme has fixed addressing with names consisting of at most $O(D\Delta)$ bits and constant adaptation cost (only the routing tables of two neighboring nodes need to be adjusted at every insertion). The space requirement for each routing table is $O(D\Delta^2)$ bits.

## 2.3 Path Labeling Scheme

There are several variants of the basic Prefix Routing scheme. One of them is as follows. As the addresses of the nodes can be quite long, it would be nice if we could shorten them as the message is being routed. The idea is to strip off a prefix of the destination address as the message passes through a node. The destination address can get shorter and shorter until eventually it becomes $\epsilon$. Thus the routing is done

as follows. If a node receives a message with a destination address that is its own address or an empty address ($\epsilon$) then it knows the message is for it. Otherwise, it tries to strip off its own address from the destination address *and* then also the maximum link label that is a prefix of the remaining address. It then routes the message with the reduced address via that link to the next node, as before. The procedure *SEND* can be modified as follows.

**Procedure** *SEND2(source,dest,m)*
{ $m$ is a message to be sent from node *source* to node *dest.*
\ denotes the operation of prefix stripping, i.e. $a \setminus b = x$, if $a = bx$ else $a \setminus b = a$. }
    **if** *source* = *dest* or $\epsilon$
    **then** process $m$
    **else**
        *dest* := *dest* \ *source* ; { strip off *source* from prefix of *dest* if possible }
        find maximum link label $x$ at *source* such that $x$ is a prefix of *dest* ;
        *source* := label of node arrived at over link $x$ ;
        send $m$ over the link labeled $x$ ;
        *dest* := *dest* \ $x$ ; { strip off the link label too }
        *SEND2(source,dest,m)*
    **endif**
**end.** { *SEND2* }

Similar to Prefix Labeling scheme, we can define a *Path Labeling scheme* as a labeling scheme that allows us to use the procedure *SEND2* and a *Path Routing scheme* as a valid Path Labeling scheme if we use procedure *SEND2.*

The idea of Path Labeling approximates the practice of specifying a path by the sequence of names of the consecutive nodes and links along the path. It appears that we only need to slightly modify the Tree Labeling scheme to obtain a *Path Labeling scheme.* Again assume that $\Sigma$ has at least two symbols: *0* and *1.* Then the nodes are labeled just as before, but the link labels are changed. The link from a son to its father in the underlying spanning tree is still labeled by $\epsilon$, but the downward link is now labeled by a sequence of *0*'s appended by a *1.* The number of *0*'s is specified by the counter value. The label of a bypass link from an ancestor down to a descendant also needs to be modified accordingly, by stripping off the ancestor's address from the descendant's name. Otherwise, everything remains identical. Formally, the procedure *INSERT2* for a Path Labeling scheme is given as follows.

**Initialization:**
    Pick a node $u$ and label it *1* ;
    c($u$) := 1
**end;**

**Procedure** *INSERT2(v,u)*
{ Insert a link from node $v$ to node $u$ . Node $u$ has name $\alpha(u)$
and a counter value of c($u$) initialized at 1. *l(v,u)* is the label for link *(v,u)* }
**if** $v$ is a new node
**then**
$\alpha(v) := \alpha(u)0^{c(u)}1$ ;
*l(v,u)* := $\epsilon$ ;
*l(u,v)* := $0^{c(u)}1$ ;
c($v$) := 1 ;
c($u$) := c($u$) + 1 ;
**else** { bypass link }
*l(v,u)* := $\alpha(u) \setminus \alpha(v)$
*l(u,v)* := $\alpha(v) \setminus \alpha(u)$
**endif**
**end.** { INSERT2 }

**Theorem 2.4** *There is a valid Path Labeling scheme for any dynamic growing network.*

**Proof.** Suppose node $u$ needs to send a message to node $v$. Again there are 3 cases:

1. *u is an ancestor of v* : Then it is possible for $u$ to strip off its address from the address of $v$, as $\alpha(u)$ is a prefix of $\alpha(v)$. Note that the reduced address cannot have *1* as a prefix. Now, there must be a link with a unique non-$\epsilon$ label that is the prefix of the reduced address, $\alpha(v) \setminus \alpha(u)$, in the spanning tree or, a bypass link if $u$ is directly connected to a descendant that is an ancestor of $v$ or to $v$ directly with a longer matching prefix as a label. After stripping off this link label, the destination address gets shorter again but still cannot have *1* as a prefix. This condition is true for every node that is on the path from $u$ to $v$ in the spanning tree until the reduced address becomes $\epsilon$. The message has then arrived at its destination.

2. *u is a descendant of v* : Then it is not possible for $u$ to strip off its address from $\alpha(v)$, since $\alpha(u)$ is longer than $\alpha(v)$ and thus not a prefix. Thus the destination address of $v$ remains intact with *1* as a prefix. None of the links in the spanning tree will have a label that is a prefix of $\alpha(v)$ because they all have *0* as a prefix, except the $\epsilon$-link to the ancestor. Thus the message will follow the $\epsilon$-link up the tree in the abscence of a bypass link. If there is a bypass link up to $v$ directly or to some ancestor of $v$ higher up the tree that matches the prefix of $\alpha(v)$ then the strip-off process applies. The address is reduced and does not have *1* as a prefix anymore. From there we get Case 1 again.

3. *Neither of the above cases* : Then $v$ belongs to a different subtree in the spanning tree, so the message will be routed up the spanning tree as in Case 2 until an ancestor or a bypass link to $v$ or to an ancestor of $v$ is encountered. Finally the path as in Case 1 will be followed.

□

There is another variant of Prefix Labeling schemes. Instead of tagging the number of *0*'s after each name in all the given schemes, one could use the counter value itself (e.g. in decimal). But then we need to increase the alphabet to include { *1, 2, ...* } and separate the counter numbers somehow, say with the symbol *!* (as in UUCP). It is then straightforward to implement a more compact form of the Tree Labeling scheme or the Path Labeling scheme. First label the root with *!*. The first son will be labeled with *!1*, the second son *!2* and so on. The sons of *!3* will be named *!3!1*, *!3!2* and so on. In short, we only have to change three lines in the insertion procedure for the Path Labeling scheme: (a) the first line of the Initialization becomes *Pick a node u and label it !*, (b) in Procedure *INSERT2*, change the statement $\alpha(v) := \alpha(u)0^{c(u)}1$ to $\alpha(v) := \alpha(u)!c(u)$ and (c) change the statement $l(u,v) := 0^{c(u)}1$ to $l(u,v) := !c(u)$. Observe that every name has a *!* as a prefix and so do the link labels except for the $\epsilon$-links. It is easy to see that it gives a valid Prefix Labeling scheme.

The above scheme has fixed addressing and constant adaptation cost. The size of the addresses can still be $O(D \log \Delta)$ bits for some graphs, but the labels are now only $O(\log \Delta)$ bits for the tree links and can be $O(D \log \Delta)$ bits for the bypass links.

## 2.4 Strict Labeling Scheme

Not all Prefix Routing schemes are necessarily Tree Labeling schemes. For example, in a fixed complete graph a node can be prefix-labeled by any arbitrary name as long as the links are labeled with the corresponding names of the nodes it is connected to. Suppose we are given an arbitrary Prefix Routing scheme on a network and now wish to insert a node at an arbitrary point. The naive scheme of picking a name for $v$ and append it to the name of $u$, as in the Tree Labeling scheme, will not work. There is no guarantee that this new found name is unique, even if $u$ knows all the names of its direct neighbors. For instance, suppose $v$ has as name $\alpha(u)x$ and $\alpha(u) = ab$ . There may be a node somewhere in the network with name $abx$ which happens to clash with the name of $v$. One way to get around this problem is to impose the following extra condition on the Prefix Labeling scheme.

**Definition 2.5** *A Prefix Labeling scheme for a network G is* Strict *if no names are prefixes of each other.*

Thus in a Strict Prefix Routing scheme the names of the nodes together form a *prefix code.*

**Lemma 2.6** *There is a scheme for insertion of nodes and links to a Strict Prefix Routing scheme such that the scheme remains Strict.*

**Proof.** Assume that $\Sigma$ contains at least *0* and *1*. Let the name of $u$ be $\alpha(u)$ and suppose $v$ is to be connected to $u$. Then we can label $v$ with $\alpha(u)1$ and relabel $u$ with $\alpha(u)0$. The link $(v,u)$ is labeled with $\epsilon$ and $(u,v)$ with $\alpha(v)$ as in the Tree Labeling scheme. The resulting scheme remains strict since the original name of $u$ is not a prefix of any other name and the current names of $u$ and $v$ are not a prefix of each other either. Any message to the new node $v$ will be routed via $u$ because $\alpha(v)$ has as prefix the old address of $u$. However the current name of $u$ has an extra *0*, so whenever $u$ receives a message it needs to strip off all the trailing *0*'s from the destination address and also its own name, then compare them to see if they are really the same. Note that there cannot possibly be two names that are identical even after all the trailing *0*'s are stripped off, otherwise they are prefix of each other to begin with. With this slight adjustment, it is easy to see that the new scheme will route messages properly and is thus valid. □

Note that the Tree Labeling scheme as given before is not Strict. But if we append a *1* to the name of each node but *not* the link labels, then it becomes Strict, since each address will then terminate with a *11* and that is the only place in the address that actually has two consecutive *1*'s. Thus, there is a Strict Prefix Routing scheme with fixed addressing and constant adaptation cost for any dynamically growing network.

# 3 Optimum Prefix Routing Scheme

## 3.1 Tree Networks

The algorithms for constructing a Prefix Routing scheme as given in the last section do not take the possibly different costs of the links into account. Thus in general, they do not give *optimum* routing schemes, i.e. messages may not follow minimum cost routes. The algorithms clearly give optimum routing for trees, as the paths are then unique. In fact, the Path Labeling scheme does give an optimum Prefix Routing scheme for trees with constant adaptation cost and each link requires only $O(\log \Delta)$ bits. However, the address of a node can still be $O(D \log \Delta)$ bits long for some trees. Thus for a chain of $n$ nodes one can have an address size of $O(n)$. Can this be improved? Unfortunately, this is not the case, as shown by the following result.

**Lemma 3.1** *Any Prefix Routing scheme for a tree with diameter $D$ requires an address size of $\Omega(D)$ for some nodes.*

**Proof.** Consider a path of length $D$ in the tree. Stretch it out as a line of $D$ nodes with $w$ as the center node. Let the nodes to the left of $w$ be called $u_1, u_2, ..., u_k$ and

those to the right $v_1, v_2, ..., v_m$ , where $k$ and $m$ are approximately $\frac{D}{2}$. Any label $w_{left}$ for the left link of the node $w$ can be extended to $p(u_1, ..., u_k)$, the maximum common prefix of the labels of all the nodes to the left of $w$, without affecting the validity of the routing scheme. Thus without loss of generality, we can assume that the left link label $w_{left}$ is equal to $p(u_1, ..., u_k)$ and that similarly, the right link label $w_{right}$ is equal to $p(v_1, ..., v_m)$. By definition, $w_{left} \neq w_{right}$. Pick the longer prefix of the two labels, say $w_{left}$. Then $w_{left}$ must contain at least 1 symbol and all the nodes to the left must have $w_{left}$ as a prefix. Now consider $u_2$, the second node to the left of $w$. Again without loss of generality, the label for the left link of $u_2$ is $u_{2left} = p(u_3, ..., u_k)$ and the label for the right link is $u_{2right} = p(u_1, x, v_1, ..., v_m)$. Now, $u_{2right} \preceq w_{right}$ but $u_{2left} \succ w_{left}$, otherwise any message to $u_1$ from $u_2$ would be routed to the left. This means that all the labels of the nodes to the left of $u_2$ must contain $u_{2left}$ as a prefix and thus must have at least 2 symbols. Consider in turn every other nodes to the left, $u_4, u_6, ...$ and repeating this argument, it follows that these nodes must have labels of increasing length (in the given order). The leftmost node $u_k$ must have a label with at least $\frac{k}{2}$ symbols, thus is of size $\Omega(D)$. □

Thus for a chain of $n$ nodes, we must have $\Omega(n)$ symbols in the address of some nodes and the Tree Labeling scheme in the last section provides this lower bound. For a balanced tree with diameter $D = O(\log n)$ and maximum degree $\Delta = O(\frac{n}{\log n})$ the address size given by the Path Labeling scheme is only $O(\log^2 n)$. Also any complete $\Delta$-ary tree has about $\Delta^D$ nodes, thus requiring $\Omega(D \log \Delta)$ bits for the names of some nodes already. Thus the Path Labeling scheme actually gives an optimum address size labeling and optimum routing for such a tree. Are there any other kind of networks that have optimum Prefix Routing schemes?

## 3.2 Fixed Networks

We first consider a restricted version of the optimality question for Prefix Routing where the topology of the network is *fixed*, i.e. there is no further insertion or deletion of nodes and links. However, we will allow that the cost of the links can vary over time. Even in this restricted case, optimality is not achievable with Prefix Routing for certain kind of networks. In this section we will explore this interesting problem and obtain a **characterization theorem** that classifies the fixed networks that do admit fully optimal Prefix Routing schemes.

**Lemma 3.2** *Any ring network* $\mathbf{C_n}$ *of size* $n \geq 5$ *has no optimum Prefix Routing scheme.*

**Proof.** Consider first the ring of size 5 labeled as in **Figure 1.**

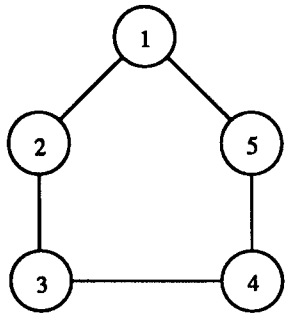


**Figure 1**

Assume all links have unit cost and assume that there is an optimum Prefix Routing scheme for this ring. We use the following notations: *p(i,j)* denotes the maximum common prefix of the labels of nodes i and j, and $a \succ b, c, \ldots$ means that each of the strings *b, c,...* is a strict prefix of string *a*. Now consider node 1. It needs to route messages to nodes 2 and 3 via link (1,2) and to nodes 4 and 5 via link (1,5). Without loss of generality we may assume that link (1,2) is labeled with p(2,3) and similarly link (1,5) with p(4,5). Since messages to nodes 2 and 3 cannot be routed via link (1,5), i.e. without using the link label p(4,5), it must be that

$$\begin{array}{cll} & 1(a): & p(2,3) \succ p(2,4),\ p(2,5),\ p(3,4),\ p(3,5) \\ \text{or} & 1(b): & p(4,5) \succ p(2,4),\ p(2,5),\ p(3,4),\ p(3,5)\ . \end{array}$$

And considering all the other nodes in turn gives us the following:

$$\begin{array}{llcll} 2(a): & p(3,4) \succ p(4,5) & \text{or} & 2(b): & p(1,5) \succ p(4,5)\ , \\ 3(a): & p(4,5) \succ p(1,5) & \text{or} & 3(b): & p(1,2) \succ p(1,5)\ , \\ 4(a): & p(1,5) \succ p(1,2) & \text{or} & 4(b): & p(2,3) \succ p(1,2)\ , \\ 5(a): & p(1,2) \succ p(2,3) & \text{or} & 5(b): & p(3,4) \succ p(2,3)\ , \end{array}$$

where we have shown only those labels that we shall need in the proof.

Now assume 1(a) is the case. Then we have $p(2,3) \succ p(3,4)$, which implies 5(b) is not the case, so we have 5(a): $p(1,2) \succ p(2,3)$, which in turn makes 4(b) impossible. Thus we have 4(a) and in turn 3(a) and then 2(a). Summarizing, we now have:

$$p(2,3) \succ p(3,4) \succ p(4,5) \succ p(1,5) \succ p(1,2) \succ p(2,3)\ ,$$

which is a contradiction. Similarly it is not possible that 1(b) is true. Hence a ring of size 5 cannot possibly have an optimum Prefix Routing scheme.

If the size of a $C_n$ is an odd number $n > 5$, then we can insert half of the number of nodes exceeding 5 between nodes 2 and 3, and the other half between nodes 4 and 5, all with unit cost links. The above argument will then cover this situation also. If n is 6, then we can insert node 6 between nodes 4 and 5. Assign the cost of $\frac{1}{2}$ each to link(4,6) and link(5,6). Then each of the five original nodes still has to route messages the same way, so the argument for the size 5 case applies. The case of rings of even size $> 6$ can be handled the same way as for odd sizes. □

Thus even with fixed link costs a ring of size 5 or more can have no optimum Prefix Routing scheme. There are networks that contain only cycles of size 4 and have no optimum Prefix Routing scheme also.

**Lemma 3.3** *The complete bipartite graph* $\mathbf{K_{2,3}}$ *with dynamic link costs has no optimum Prefix Routing scheme.*

**Proof.** Let the network be given as in **Figure 2.**

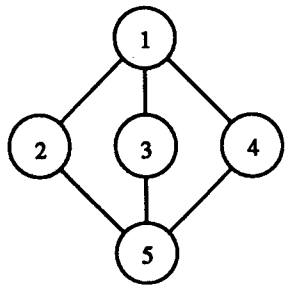


**Figure 2**

Call a link (i,j) of any network *forbidden* if no optimum path from i to j includes it. It is clear that any optimum path in the network cannot contain a forbidden link. Assume there is a labeling scheme such that the Prefix Routing scheme is optimum. Consider node 3. Suppose the forbidden links are link (1,2) and link (4,5). Then node 3 must route messages to nodes 1 and 4 via link (1,3), and nodes 2 and 5 via link (3,5). As before, we assume that link (1,3) and link (3,5) might as well be labeled with p(1,4) and p(2,5), the maximum common prefix of the corresponding nodes, respectively. Since messages to nodes 1 and 4 cannot be routed via link (3,5) using the label p(2,5), it must be that

$$\begin{array}{lll} & 3(a): & p(2,5) \succ p(1,2),\ p(2,4),\ p(1,5),\ p(4,5) \\ \text{or} & 3(b): & p(1,4) \succ p(1,2),\ p(2,4),\ p(1,5),\ p(4,5). \end{array}$$

Suppose the forbidden links now become link (1,4) and link (2,5), as the costs of the links vary over time. Since the addresses are fixed we can no longer change the names of the nodes. By the same reasoning as above, we must have

$$\begin{array}{lll} & 3(A): & p(4,5) \succ p(1,4),\ p(2,4),\ p(1,5),\ p(2,5) \\ \text{or} & 3(B): & p(1,2) \succ p(1,4),\ p(2,4),\ p(1,5),\ p(2,5). \end{array}$$

If case 3(a) is true, then we have $p(2,5) \succ p(4,5)$ and this contradicts 3(A): $p(4,5) \succ p(2,5)$. So it must be 3(B), which contains $p(1,2) \succ p(2,5)$, but this is impossible due to 3(a). Thus 3(a) cannot be the case. Similarly neither can 3(b). So there is no optimum Prefix Routing scheme for the network under the given constraints. □

Combining the above two Lemmas, we have the following theorem.

**Theorem 3.4** *If a network with dynamic link costs has an optimum Prefix Routing scheme then it cannot contain any subgraph with a cycle of length* $> 4$ *or a* $\mathbf{K_{2,3}}$.

Before we state the positive results, some preliminaries are needed. We shall use a construction similar to tree labeling based on some spanning tree, except that each node of the tree may comprise a special group of nodes. Let us call each group node a $g$-node . Each $g$-node will have one of its member nodes designated as the root node. Recall that in the construction of the Tree Labeling scheme in section 2, each descendant node was given an $\epsilon$-labeled link connected to its direct ancestor. Thus in order for the spanning tree with $g$-nodes to route messages properly, each member of a $g$-node should have an $\epsilon$-link also. There are basically two sizes of $g$-nodes : those with 3 or 4 elements. The following lemmas show why they are important for the scheme.

**Lemma 3.5** *There is an optimum Prefix Routing scheme for a fixed biconnected component of 4 nodes with dynamic link costs such that every node in it (except the root node) has an* $\epsilon$*-link.*

**Proof.** Label the root node $a$, for some non-$\epsilon$ string $a$, and the other 3 nodes $ax$, $ay$ , and $az$, where $x$, $y$ and $z$ are distinct, and the maximum common prefix of $x$, $y$ and $z$: $p(x,y,z)$ is not $\epsilon$. Note that any biconnected component of 4 nodes can be obtained from the graph in **Figure 3** by assigning prohibitive cost to appropriate links to make them forbidden.

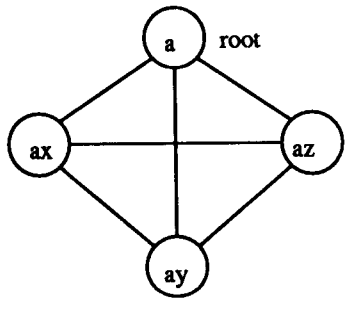


**Figure 3**

Suppose costs have been assigned to the links. Determine a set of shortest paths $P$ such that every subpath of a path in $P$ is also in $P$ and for any two nodes $u$ and $v$ there is a unique shortest path in $P$ from node $u$ to $v$. Any forbidden link is labeled by a string $b$ that is not a prefix of any of the labels for the nodes. We label the links of a node $u$ according to the number of forbidden links that it has, as follows.

1. *There are 2 forbidden links*: So the only path in $P$ left for $u$ is via link $(u,v)$ for some $v$. If $u$ is the root node then label link $(u,v)$ with $p(ax,ay,az)$ else label it with $\epsilon$ at $u$.

2. *There is only 1 forbidden link*: So there are two paths in $P$ that $u$ can use. One path, say via link $(u,v)$, is of length 1 and the other path, say via link $(u,w)$, must be of length 2. We label the links using the following routine.

```
if u is the root node
then
      l(u,v) := α(v) ;          { label link (u,v) with address of v }
      l(u,w) := p(ax,ay,az)
else
      if v is the root node
      then
            l(u,v) := ε ;
            l(u,w) := p(ax,ay,az)
      else
            l(u,v) := α(v) ;
            l(u,w) := ε         { root node is along this path }
      endif
endif.
```

3. *There is no forbidden link*: Then $u$ has three available links. If node $v$ is the root node then label link $(u,v)$ with $\epsilon$ otherwise label link $(u,v)$ with $\alpha(v)$.

We claim that the above routine assigns labels correctly and optimally. Suppose node $u$ sends a message to node $v$. It is easy to check that if node $v$ is of path length 1 away from $u$ in the shortest path set $P$ then the above routine routes the message in one hop also. This gives us the correct induction basis. Now assume the routine routes messages correctly for path length $k \geq 1$ and the path length from $u$ to $v$ is $k+1$. Then node $u$ cannot be of type 3 above. So node $u$ must have 1 or 2 forbidden links. If it is of type 1 then there is only one link available and the link label is such that $\alpha(v)$ satisfies the prefix requirement. If it is of type 2 then there are 2 links available, but the link labels are such that $\alpha(v)$ satisfies only the prefix requirement for the link with path length 2. In either case the message is routed to the next node via the correct link and by induction hypothesis correctly and optimally from there on.
If the link costs change, then the link labelings can be changed by only local alterations to keep the scheme optimum. □

**Lemma 3.6** *There is an optimum Prefix Routing scheme for a biconnected component that is a fixed* $\mathbf{C_3}$ *(a cycle of 3 elements) with dynamic link costs such that every node in it (except the root node) has an $\epsilon$-link.*

**Proof.** Let the root node of the $\mathbf{C_3}$ be labeled $a$ where $a$ is not $\epsilon$ and the other two nodes $ax$ and $ay$, where $x$ and $y$ are distinct and the maximum common prefix of $x$ and $y$: $p(x,y)$ is not $\epsilon$.

Let a cost be assigned to each link. The forbidden link at any node is labeled with a string that is not a prefix of any of the labels for the nodes. For any node $u$ we label the links as follows.

1. *u has no forbidden link*: Then label any link $(u,v)$ at $u$ with $\epsilon$ if node $v$ is the root node, otherwise with $\alpha(v)$.

2. *u has one forbidden link*: Then the only available path left is via link $(u,v)$ for some $v$. If $u$ is the root node then label link $(u,v)$ with $p(ax,ay)$ else label it with $\epsilon$.

It is now routine to verify that the above labeling scheme gives a valid and optimum Prefix Routing scheme for the component.
In the case of link cost changes, the link labeling is easily updated accordingly. □

**Theorem 3.7** *Any fixed network with dynamic link costs which contains no cycle of length greater than 4 and no subgraph* $\mathbf{K_{2,3}}$ *has an optimum Prefix Routing scheme.*

**Proof.** Consider any (connected) network $G$ that contains no (simple) cycles of length $> 4$ and no $\mathbf{K_{2,3}}$. It is easy to verify that if a biconnected component of size 5 or more contains only cycles of length 4 or less then it must contain the subgraph $\mathbf{K_{2,3}}$. Thus biconnected components of $G$ are necessarily of size 4 or less, and can be viewed as $g$-nodes, with the remaining edges of $G$ connecting them as a spanning tree. After choosing a root of this tree, assign the 'root-nodes' in every $g$-node and label the network by the technique used in the Tree Labeling scheme. Upon encountering a biconnected component of size 3 or 4, label the links inside the $g$-node as in Lemma 3.6 or Lemma 3.7. Note that the labels for the nodes satisfy the conditions in the two lemmas.

We claim that the above routing scheme is optimum. Suppose node $u$ sends a message to node $v$. If $u$ and $v$ are in the same component, i.e., $g$-node, then by Lemma 3.6 and Lemma 3.7 the message is routed optimally. So suppose they are in different $g$-nodes. Call a $g$-node $\alpha$ a *group-ancestor* of $g$-node $\beta$ if the path from the root node to $\beta$ passes through $\alpha$. We then also call $\beta$ a *group-descendant* of $\alpha$. Observe that if $v$ is in the group-descendant of the $g$-node $u$ then $v$ must be a proper descendant of some node $w$ in the $g$-node $u$, hence $\alpha(w)$ must be a prefix of $\alpha(v)$. So a message to $v$ must be routed through $w$ and from there by induction hypothesis it will be routed optimally. Similarly, if $v$ is in the group-ancestor of the $g$-node $u$ then the message will be routed through the local root node via the $\epsilon$-link. If neither of the above is the case then the only link label that will satisfy is the $\epsilon$-link. The message will travel via $\epsilon$-links until it reaches a group-ancestor of $g$-node $v$ and then will travel to $v$ as in the first case.

As the costs of the links change so do the routing tables of the adjacent nodes, according to Lemma 3.6 and Lemma 3.7. Since the biconnected component are only

of size 3 or 4 the adaptation cost is of $O(1)$. The routing tables for nodes that do not belong to a biconnected component (the *tree* nodes), need not be changed at all. □

Combining the above results, we obtain a characterization of the fixed networks with dynamic link costs that have an optimum Prefix Routing scheme.

**Theorem 3.8 (Characterization)** *A fixed network with dynamic link costs has an optimum Prefix Routing scheme iff its biconnected components are of size 4 or less.*

## 3.3 Dynamic Networks

In a fully dynamic network, nodes and links can be inserted and deleted arbitrarily. Since we require our network to be connected, we do not allow deletion of links that will cause disconnection of the network. It is clear from the construction of the schemes for biconnected components of size 3 and 4 that the links of the components can even be deleted, as long as this causes no disconnection of the network. This is so, because deletion tantamounts to levying a prohibitive cost on that link. Insertion of nodes with one link can be done without affecting the optimality of the scheme. So is the insertion of links such that the cycles formed are of size 3.

**Theorem 3.9** *Any dynamic network with arbitrary insertion and deletion of nodes and links, such that there is no disconnection of the network and no cycle of length greater than 3 is formed at any time, has an optimum Prefix Routing scheme.*

**Proof.** Consider the network at some moment, with an optimum Prefix Labeling in effect according to the Tree Labeling scheme. There are only two ways that a cycle of length 3 can be formed dynamically.

1. *A node $u$ is connected to one of its siblings $v$* : In which case both $u$ and $v$ have a common father, say with node label $a$. Then $u$ must have label $ax$ for some non-$\epsilon$ string $x$ and similarly $v$ must have label $ay$ where $y$ is also non-$\epsilon$ and distinct from $x$. The condition for Lemma 3.7 is satisfied, so the routing tables of the three nodes can be adjusted accordingly to be optimum.

2. *A grandson node $u$ is connected to its grandfather $v$*: Then similarly $v$ must be labeled by some $a$, $u$'s father $w$ by $ax$ and $u$ by some $ay$, where $x$ is a strict prefix of $y$. Again the condition for Lemma 3.7 is satisfied.

□

The above construction does not work if a cycle of length 4 is formed arbitrarily. In fact, we have the following.

**Lemma 3.10** *There is no optimum Prefix Routing scheme for dynamic networks of more than 4 nodes with dynamic link costs that allows arbitrary insertion of nodes and links such that cycles of length 4 are formed.*

**Proof.** Suppose we have at least 5 nodes and that eventually the connections as in **Figure 4** are made.

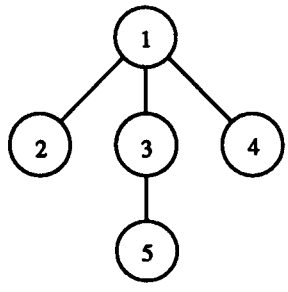


**Figure 4**

Then there are two ways in which a cycle of length 4 can be formed.

1. *link (2,5) is connected*: Then this is case 1 of $\mathbf{K_{2,3}}$ in Lemma 3.4, if we consider link (4,5) as forbidden.

2. *link (4,5) is connected*: This is case 2 of $\mathbf{K_{2,3}}$ in Lemma 3.4, if we consider link (2,5) as forbidden.

In either case we have a contradiction as in Lemma 3.4. So it is not possible to label the graph properly so a robust, optimum Prefix Routing scheme is obtained. □

Note that the above argument does not contradict the result for cycles of length 3, because we then get the situation as in the case of biconnected components of size 4, which can be done according to Lemma 3.6. We summarize the above results in the following theorem.

**Theorem 3.11 (Characterization)** *A dynamic graph of more than 4 nodes with dynamic link costs that allows arbitrary insertion and deletion of nodes and links, such that there is no disconnection of the network, has an optimum Prefix Routing scheme iff it contains no cycle of length* $> 3$.

# 4 Hierarchical Prefix Routing Scheme

The Prefix Routing schemes introduced so far assume only one label per link. If we relax this requirement and allow an arbitrary number of labels (up to $n$) per link then we obtain a so called *multi-label Prefix Routing scheme.*

**Definition 4.1** *A Routing scheme is called a* $k$-Prefix Routing scheme *(or* $k$-PRS *for short), where* $k \geq 1$, *if it employs the prefix routing technique and there are only at most* $k$ *labels per link.*

Thus a 1-PRS is the standard Prefix Routing scheme. It is clear that the set of networks with dynamic link costs that have an optimum $k$-PRS is contained in the set of networks with dynamic link costs that have an optimum $(k+1)$-PRS (denotation: $k\text{-}\mathcal{PRS} \subseteq (k+1)\text{-}\mathcal{PRS}$), so that we have a hierarchy of Prefix Routing schemes. As before we implicitly assume that only valid $k$-Prefix Routing schemes are considered.

In the last section, we saw that a ring of size $> 4$ has no optimum 1-PRS. However, the corresponding situation is more pleasant if we allow a 2-PRS.

**Lemma 4.2** *Any fixed ring with dynamic link costs has an optimum 2-PRS.*

**Proof.** Label the nodes *1, 101, 1001, ...* (say) clockwise around the ring and assume some initial assignment of link costs. For each node $u$, find the minimum spanning tree with $u$ as the root. Assume clockwise is right and counterclockwise is left. Let the leftmost and rightmost leaves of the tree be $u_l$ and $u_r$ respectively (if they exist); also, let $v$ be the next neighbor to the right (clockwise) from $u$. Then the links of $u$ are labeled as follows: If there is no leftmost or rightmost leaf then label the forbidden link *0* ( since no node has such a prefix) and the other link $\epsilon$. Otherwise, label the right link of $u$ by $v$ and the left link $u_l$. Whenever the rightmost leaf $u_r$ is *not* a prefix of $u$ (i.e. the tree contains node *1* ) we also need to double-label the right link with $\epsilon$. Similarly, we double-label the left link by $\epsilon$ if the leftmost leaf is not a prefix of $u$. Fortunately, these two conditions cannot occur simultaneouly, otherwise we would have two $\epsilon$'s on the links. It can be shown by induction that this indeed yields an optimum scheme.
Note that in the case of link cost changes it is possible that almost the entire link labeling of the ring network has to be updated in order to retain an optimum scheme.
□

In a fixed network, it has been shown in [SK85, vLT86, FJ88] that any ring has an optimum Interval Routing scheme. Thus in the static case, there seems to be some connection between 2-PRS and 1-IRS. It turns out that indeed the last lemma is a special case of the following general condition. For the notion of $k$-IRS see [vLT87].

**Theorem 4.3** *Any fixed network that has an optimum* $k$*-IRS also has an optimum* $(k+1)$*-PRS.*

**Proof.** Let $G$ be a network that admits a $k$-IRS, with node labels *1, 2, 3, ...* . Then we can relabel the nodes *1, 101, 1001, ...* . For each link-label $j$ we relabel it with a *1* followed by $(j-1)$ *0*'s. If there is a special link-label $j > 1$ that contains *1* as an interval (i.e. there is a wraparound) then we need to label that link with $\epsilon$ also.

Let the network so labeled be called $G'$. Then there is a natural correspondence between $G$ and $G'$: *node label* $a \leftrightarrow 10^{a-1}1$ *if* $a \neq 1$ and *link label* $j \leftrightarrow 10^{j-1}$. Note that the link labels satisfy the following property: $i \leq j$ in $G \leftrightarrow i' \preceq j'$ in $G'$, where $i'$ and $j'$ are the corresponding links for $i$ and $j$, and $\preceq$ is the prefix relation.

We claim that the above translation of schemes is correct. For suppose that node $a$ sends a message to node $b$ using the $k$-IRS in $G$. Then node $a$ looks up its routing table and find the maximum link label $i$ such that $i \leq b$. It then routes the message via that link. In the corresponding $G'$, node $a'$ will look up its routing table also and find link label $i'$ where $i' \preceq b'$. If there is a wraparound, i.e. there is no link label $i$ in $G$ such that $i \leq b$, then node $a$ will pick the maximum label $i$ and that interval will contain *1*. But in this case that link will have an extra label in $G'$ labeled $\epsilon$, so this will be the only link that will satisfy the prefix requirement in $G'$. In either case both $G$ and $G'$ will route message using the same link. Thus if $G$ has an optimum and valid $k$-IRS then $G'$ has an optimum and valid $(k+1)$-PRS also. □

Note that in the above construction, only a single link per node requires an extra label and this causes the $k$-PRS to jump up to $(k+1)$-PRS; the rest of the links have at most k labels. This is because the idea of wraparound the interval is not inherent in the Prefix Routing scheme, so we need an extra $\epsilon$-labeled link to handle this special case.
On the other hand we have the following lemma.

**Lemma 4.4** *There exists a fixed network with dynamic link costs that has an optimum* 1*-PRS but no optimum* 1*-IRS.*

**Proof.** Let the network be given as in Figure 5.

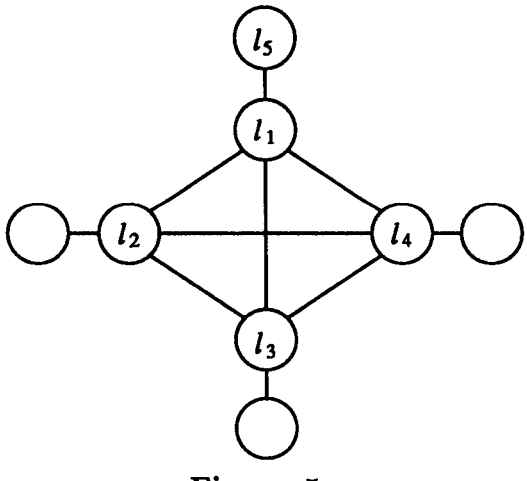


**Figure 5**

Clearly all the biconnected components of this network are of size 4 or less, hence by theorem 3.8 it has an optimum 1-PRS.

Now we will show that there doesn't exist an optimum 1-IRS for this network with dynamic link costs. Assume that the names $l_1, l_2, l_3, l_4$ and $l_5$ are given to nodes of the network as indicated by figure 5. W.l.o.g. we may assume that $l_1 < l_2 < l_3 < l_4$. Assign link costs such that the links $(l_1, l_4), (l_2, l_3), (l_3, l_4)$ become forbidden links. To route messages from node $l_1$ to nodes $l_2$ and $l_4$ optimally the label of link $(l_1, l_2)$ must be an interval containing the interval $[l_4, l_2]$. As messages from node $l_1$ to node $l_5$ will not be routed over link $(l_1, l_2)$, it follows that $l_5 \notin [l_4, l_2]$. Hence we have two possible cases: 1) $l_2 < l_5 < l_3$ or 2) $l_3 < l_5 < l_4$. We will show that neither of the two possibilities will allow an optimum 1-IRS.

1. $l_2 < l_5 < l_3$: Assign link costs such that links $(l_1, l_3), (l_2, l_4)$, and $(l_3, l_4)$ become forbidden links. To route messages from node $l_1$ to nodes $l_2$ and $l_3$ optimally link $(l_1, l_2)$ must be labeled with an interval containing the interval $[l_2, l_3]$. But $l_5 \in [l_2, l_3]$, hence this scheme is not valid.

2. $l_3 < l_5 < l_4$: By a very similar argument one shows that there cannot be an optimum 1-PRS.

□

Thus there is no containment relation between 1-$\mathcal{PRS}$ and 1-$\mathcal{IRS}$. In [FJ88] a slightly different definition of 1-IRS is used. Here it is required that each link of a node is labeled with an interval that doesn't contain the label of the node itself. We denote this version with 1-IRS'.
In [FJ88] it has been shown that a network with dynamic link costs has an optimum 1-IRS' iff the network is outerplanar. It is known that a graph is outerplanar iff it does not contain a subgraph that is a subdivision (i.e. by adding zero or more nodes into the links of the graph) of $\mathbf{K_{2,3}}$ or $\mathbf{K_4}$ (See [CH67]). We have seen that the complete graph of 4 nodes $\mathbf{K_4}$ with dynamic link costs has an optimum 1-PRS. Thus even without lemma 4.4 it is clear that there is no containment relation between 1-$\mathcal{PRS}$ and 1-$\mathcal{IRS}'$. On the other hand, it can be shown that 1-$\mathcal{PRS} \subset$ 2-$\mathcal{IRS}$.

**Lemma 4.5** *Any fixed network with dynamic link costs which contains no biconnected components of size* $> 4$ *has an optimum* 2*-IRS.*

**Proof.** Assume a network with no biconnected components of size $> 4$ is given. Let $n$ be the size of the network. This network can always be given a dfs-numbering such that in every biconnected component of size 4 we have the situation shown in figure 6, where $l_1, l_2, l_3, l_4$, and $s$ are dfs-numbers such that $l_1 < l_2 < l_3 < l_4 \leq s \leq n$, and $L_1 = [1, l_1) \cup (s, n] = (s, l_1)$, $L_2 = (l_2, l_3)$, $L_3 = (l_3, l_4)$, and $L_4 = (l_4, s]$ are the sets of dfs-numbers of the nodes outside this component that only can be reached through the nodes with dfs-number $l_1, l_2, l_3$, and $l_4$ respectively.

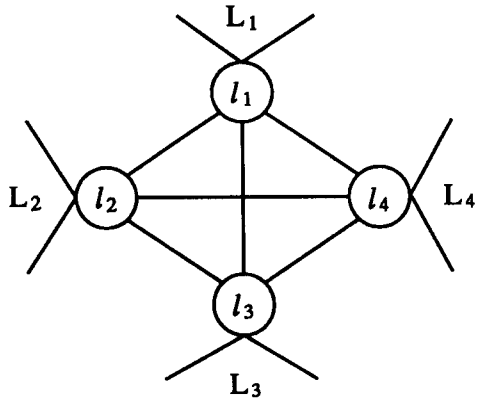


**Figure 6**

Let $v$ be a node of the component. If 0 or 2 links adjacent to $v$ are forbidden links of the component, then it is easy to see that each non-forbidden link adjacent to $v$ needs only one interval label to route messages optimally. If exactly 1 link adjacent to $v$ is forbidden, then one of the two non-forbidden links will need one interval label only whereas the other will need at most two interval labels to route messages optimally.
Similarly it can be shown that links in biconnected components of size 3 and links in the other connected components of the network need only one interval label to route messages optimally. □

# 5 Conclusion

There are several open questions. First, concerning the hierarchical Prefix Routing scheme, only 1-$\mathcal{PRS}$ is characterized. What about $k$-$\mathcal{PRS}$ for $k > 1$? Also, the containment in the hierarchy $k$-$\mathcal{PRS} \subseteq (k+1)$-$\mathcal{PRS}$ should be strict. For the containment of $k$-$\mathcal{PRS} \subset (k+1)$-$\mathcal{IRS}$, only the containment 1-$\mathcal{PRS} \subset$ 2-$\mathcal{IRS}$ is known and the proof of this depends on the characterization of 1-$\mathcal{PRS}$.

If node labels can be chosen after the assignment of link costs, the set of networks with optimum $k$-PRS ($k > 1$) will probably be larger than $k$-$\mathcal{PRS}$. No characterizations of these sets are known. (In [FJ88] a characterization of the class of networks is given in the case of 1-Interval Routing.)

Even though dynamic link costs can be considered as link deletion without disconnecting the network, we have not considered fault-tolerant networks in general. In a truly dynamic environment, nodes and links can go down and up rather arbitrarily. Can a Prefix Routing scheme be adapted for such a network?

Not all dynamic networks can have a constant adaptation cost regardless of what schemes we used. For example, a ring of size $n$ with dynamic links require updating of all the local routing tables of the whole ring. But for trees, is there a dynamic routing scheme that will give a constant (amortized) adaptation cost but with a smaller address size than $O(D \log \Delta)$ as for Prefix Routing schemes?

# 6 Acknowledgements

We thank Gerard Tel for useful comments.

# References

[ABLP89] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. *Compact distributed data structures for adaptive routing.* Proc. 21st Annual ACM Symp. on Theory of Computing, 1989, pp. 479-489.

[AGR89] Y. Afek, E. Gafni, and M. Ricklin. *Upper and lower bounds for routing schemes in dynamic networks.* To appear.

[CH67] G. Chartrand and F. Harary. *Planar permutation graphs.* Annales de l'Institut Henri Poincaré, Section B3, 1967, pp. 433-438.

[FJ86] G. N. Frederickson and R. Janardan. *Separator-based strategies for efficient message routing.* Proc. 27th IEEE Annual Symp. on Foundation of Computer Science, 1986, pp. 428-437.

[FJ88] G. N. Frederickson and R. Janardan. *Designing networks with compact routing tables.* Algorithmica, Vol. 3, 1988, pp. 171-190.

[MM88] U. Manber and L. McVoy. *Efficient storage of nonadaptive routing tables.* Networks, Vol. 18, 1988, pp. 263-272.

[NL83] D. A. Nowitz and M. E. Lesk. *A dial-up network of UNIX systems.* In Unix Programmer Manual, 7th ed., Vol. 2, Holt, Rinehart, and Winston, New York, 1983.

[PU88] D. Peleg and E. Upfal. *A tradeoff between size and efficiency for routing tables.* Proc. the 20th Annual ACM Symp. on Theory of Computing, 1988, pp. 43-52. Revised version: Journal of the ACM, Vol 36, No. 3, July 1989, pp. 510-530.

[QH86] J. S. Quaterman and J. C. Hoskins. *Notable computer networks.* Communications of the ACM, Vol. 29, 1986, pp. 932-971.

[SK85] N. Santoro and R. Khatib. *Labelling and implicit routing in networks.* The Computer Journal, Vol. 28, No. 1, February 1985, pp. 5-8.

[vLT86] J. van Leeuwen and R. B. Tan. *Computer networks with compact routing tables.* In, G. Rozenberg and A. Salomaa (eds.) The Book of L, Springer-Verlag, New York, 1986, pp. 259-273.

[vLT87] J. van Leeuwen and R. B. Tan. *Interval routing.* The Computer Journal, Vol. 30, No. 4, 1987, pp. 298-307.