Trainet: A new label switching scheme

Yehuda Afek Anat Bremler-Barr
Computer Science Department
Tel-Aviv University
Tel-Aviv 69978, Israel
{afek,natali}@math.tau.ac.il

Abstract—
***Trainet*, a new scheme to extend MPLS (Multi Protocol Label Switching) is presented. The scheme works much like the subway system in a large metropolitan area. Each (unidirectional) subway line corresponds to a labeled path, and a route in the network is defined by either a pair <label,count-value>, where count specifies how many hops a packet still has to take in the specified train, or a route may be defined by a sequence of such pairs. A sequence of such pairs specifies that the packet has to take a number of hops in one train-line, and then continue for a certain number of hops on another train-line and so forth. While slightly increasing the number of labels in a header and adding a counter to each label, the scheme reduces the total number of different labels necessary in the network, and in each switch. Thus, for a given number of labels it may support a larger number of flows. Moreover, our scheme considerably simplifies the path set up cost while still providing all the features of MPLS: switching, supporting QoS, explicit routing, and traffic engineering.**

I. Introduction

Combining switching technology with IP routing is a popular avenue of research and development in recent years. Supporting higher speeds of routing is not the only benefit expected from this combination. Supporting QoS forwarding/routing, explicit routing, and traffic engineering are additional benefits resulting from the combination. In this paper we offer and examine yet another variant on this combination, called *Trainet*. The new scheme builds upon and extends the MPLS protocol, improving its functionality and flexibility.

Standard IP routing is a hop by hop routing, where in each hop (router) the routing decision is based only on the destination address of the packet. At any given point in time all packets with the same destination address would be routed in exactly the same way, regardless of any other parameter of the packet. At each router along the packet path the destination address of the packet is examined and the longest prefix that matches the address, out of all the prefixes (usually several tens of thousands) in a forwarding table is found. The packet is then routed according to the information associated with the Best Matching Prefix (BMP) in the forwarding table.

The cost of computing the BMP at each router combined with the inflexibility to distinguish between different flows with the same BMP motivated the development of label switching protocols. All the packets of the same flow are tagged with a unique label upon entering the network layer. At each switch (router) on the path the label is used to switch (rather than route) the packet to its next hop. The label is used as an index into the switching table. That is, label switching supports the concept of "route once and switch many", for all the packets in a flow routing is performed once when setting up the label path, and the label is then used for all the packets at all the intermediate switches.

The large number of different label switching protocols [1] (tag-switching [2], ARIS [3], CSR [4], IP-Switching [5], and threaded indices [6]) motivated the IETF to develop MPLS, a standard approach to label switching [7]. MPLS is designed to combine the best of the different approaches.

The basic advantages of the MPLS protocol are; First, it enables gigabit and higher rates of packet processing (which is also enabled by the techniques of [8], [9], [10], [11], [12], [13], [14]). Secondly, MPLS supports QoS forwarding by assigning different labels to flows that require different services even if destined to the same destination. Thirdly, MPLS enables explicit routing (source routing) in which the source determines the entire path taken by the packets. While in IP source routing is achieved by carrying a complete explicit route in each packet, in MPLS it is enough to carry the explicit route only by the setup packet. Henceforth the forwarding is performed by switching on the label which was associated with the explicit route. The explicit routing feature of MPLS enables the support of good traffic engineering, load-balancing, and support for resource reservations. Essentially MPLS establishes a circuit between the source and destination, which in the terminology of MPLS is called Label Switched Path (LSP).

In this paper we examine a possible extension of MPLS that we call *Trainet*. The extension is based on two ideas: First we suggest to increase the utilization of each label by using it to define each sub route of a labeled switched path (LSP). It is achieved by replacing the label with a <label,counter-value> pair. At each switch the counter tells how many more hops the corresponding packet should take on this LSP. This is much like taking a subway, where the packet has to go off the train after a specified number of stations. We thus call each LSP in this method a *train-line*. The implementation of this idea requires the hardware which is necessary to decrement the counter and in parallel to check whether it was equal one. A packet that has reached a switch with the counter value equals to one has reached its destination on this train-line. The simple to implement, decrement operation is not new, it is required in any event to implement the TTL mechanism of IP (which comes to avoid infinite looping).

Notice however that the above idea supports merging of LSPs that arrive at a router on different interfaces and continue along the same path (as MPLS does). That is, in both MPLS and Trainet, if a set of LSPs or train-lines continue along the same path from some point on, then all these LSPs (or train-lines) may be merged into one LSP (train-line) from this point on.

The second idea in the extension is to specify a route in the network by a sequence of a small number of labels. The idea may be applied separately to either MPLS as is, or to the train-lines in Trainet. This is like taking a trip in a subway system, from one point to another which is combined of several train-lines. We apply this idea to either MPLS as is (without counters), or to the Trainet extension. The implementation is supported by the stack option of MPLS, i.e., placing the sequence of labels or <label,counter> pairs on each packet and performing the pop operation on the stack at the appropriate switches. As we will show the multi label route approach may be of considerable benefit even if applied only to the MPLS labels. When applied in the Trainet method it further opens many beneficial possibilities for making the label switching method more efficient.

The main advantages of the Trainet ideas are: (1) reducing the number of labels necessary in the network thus enabling more label switched routes with the same number of labels, and (2) eliminating some of the MPLS setup over-heads whenever an infrastructure of train-lines exists in the network. However, to support the Trainet ideas the following is required: (1) to replace the labels with a <label,counter-value> pair and have the necessary hardware to perform the decrement and compare with “1” operations on the counter-value. If the MPLS switch is built on an ATM hardware then this is problematic as there is no support for such mechanisms in the VC/VP of ATM. (2) to assume MPLS switches that support the pop and push operations (for the multi label route).

In Section II we give a brief overview of the MPLS protocol. The basic concept of the two ideas in this paper are given in Section III. A topology based implementation of the train-line idea is discussed in Section IV. The idea of using multiple labels in MPLS is discussed in Section V, and the usage of multi train-lines for setting up an explicit route is presented in Section VI. In Section VII it is shown how the Trainet idea may be used to quickly recover from link/router failures at any point along the LSP. Concluding remarks and future work are in Section VIII.

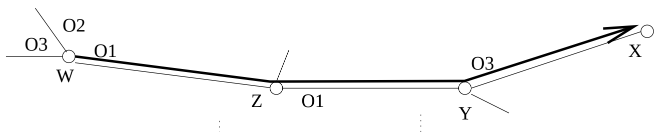
II. Short MPLS overview

Here we review some basic concepts of MPLS that are relevant to our extension. Readers familiar with MPLS should skip this section and continue in Section III. Throughout the paper we use MPLS terminology.

In MPLS all the packets that follow a certain route are labeled with the label that is assigned to this route. This label is used for switching (rather than forwarding) of packets in routers along the corresponding route. To simplify the process of coordinating the label selection, a different label value may be used at each hop along the route. The linking between the label value in one hop to the label value on the next hop is obtained via a table of labels in each switch (which is also the switching table). Upon setting up a LSP each down-stream router informs its upstream neighbor what label it has assigned to that LSP incoming link. The upstream router would place this label on each packet of the corresponding flow that it forwards to the downstream router. Each packet arriving at a Label Switch Router (LSR) with label *l-in* is forwarded according to the outgoing port (line card) found at the Incoming Label Map (ILM) at entry *l-in*. While being switched the *l-in* label is swapped with the *l-out* label that is also found in the *l-in* entry of the ILM. This procedure is called label swapping.

Each LSR (Label Swapping Router) maintains two basic tables, the Incoming Label Map (ILM) which is used for switching and the FEC (Forwarding Equivalent Class) Map which is used for routing of packets that arrive without a label. The ILM table is implemented in hardware in order to perform switching at high rates. Each entry of the ILM contains the next link on the path (interface card), the label that should be swapped into the packet according to the downstream label assignment, and possibly other fields such as an operation (to pop or push a new label on the packet). An example of a LSP is provided in figure I.

The FEC map is a forwarding table used by packets that enter the MPLS cloud. Each entry in the table corresponds to all the messages that belong to the same Forwarding Equivalence Class (FEC), i.e., that are forwarded through the MPLS cloud in the same way. Packets that belong to the same FEC get the same label and travel the same route in the entire MPLS domain. An example of a FEC could be: all the packets with the same first 20 bits in the destination address and with the same QoS requirements.



ILM for LSR Z

l-in	l-out	out-port
1		
2		
3		
...		
5	21	O1
6	14	O1
...		

ILM for LSR Y

l-in	l-out	out-port
1		
2		
...		
21	8	O3
14		End
...		

FEC-Map for LSR W

dest	l-out	out-port
X	5	O1
Y	6	O1

FEC-Map for LSR Z

dest	l-out	out-port
X	21	O1
Y	14	O1

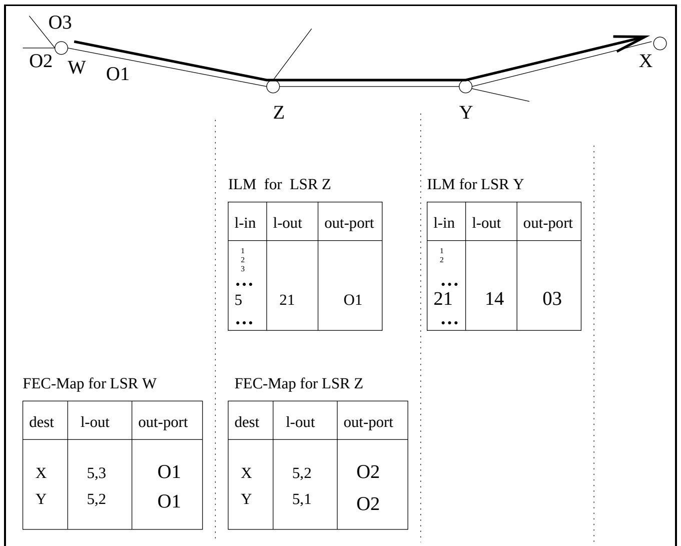
Fig. I. LSP in MPLS. Consider the case where LSR W sends packet to Y. LSR W analyzes the network header to find the packet FEC. It labels the packet with label 6, that it finds in the FEC-MAP. Z receives this packet and according to its ILM and the incoming label 6 switches the packet to interface 1 with label 14. Notice that in MPLS, the LSP from W to Y cannot benefit from the fact that a LSP between W to X exists.

The labels in MPLS are a scarce resource. First they are the key element in the scalability of MPLS, and the more labels we need the larger the ILM tables are. Since the ILM is a hardware switching table that has to operate at high speeds, it is built out of expensive memories and its size is critical. Therefore various methods to reduce the number of labels necessary in MPLS have been considered, e.g., merging LSPs which means using the same label for all the packets with the same destination even if they arrive from different ports. When there are not enough labels to distinguish between all the routes, the system uses fewer labels and thus is less discriminating between the different FECs.

Furthermore, two basic approaches were suggested for the label assignment and distribution in MPLS, Topology/Control based, and Data based. In the topology/control based approach an LSP between any source destination pair is setup ahead of time by the routing/control protocols. The disadvantage of this approach is that it requires a rather large number of labels (e.g., 60,000 in a backbone BGP label switch router). The data driven approach works "on demand". The arrival of a new flow at an LSR triggers the establishment of a new LSP by calling on a label distribution protocol (LDP). While this approach may save on the number of labels used in the network it is more expensive in the overhead of setting up and tearing down an LSP.

III. The Trainet concept

The first basic idea behind the Trainet method is to be able to use the same label to specify more routes. That is, to enable a packet that flows on a particular labeled switch path to end its traversal at any switch on the way and not only at the end of the LSP. This is accomplished by adding a counter to the label carried by each packet, which indicates how many more switches the packet should go through in this LSP. While being forwarded as in MPLS, each switch decrement the counter by one and when the counter value reaches zero the packet has reached its destination on this LSP. See for example figure II.



ILM for LSR Z

l-in	l-out	out-port
1 2 3 …		
5	21	O1
…		

ILM for LSR Y

l-in	l-out	out-port
1 2 …		
21	14	03
…		

FEC-Map for LSR W

dest	l-out	out-port
X	5,3	O1
Y	5,2	O1

FEC-Map for LSR Z

dest	l-out	out-port
X	5,2	O2
Y	5,1	O2

Fig. II. Using the train idea we eliminate the need for a separate label for the path from W to X, and a separate label for the path from W to Y, as in figure I.

To exemplify the idea consider a ring topology. In MPLS we would need n labels to support all possible routes on a ring of size n, one for each possible destination. In Trainet two labels would suffice for a ring of any size, one for the clockwise train-line and one for the counter clockwise train-line. The counter attached to each Trainet-label indicates how many hops are necessary to take in that direction in order to reach the destination (on the shortest path). In another example consider a tree topology network. In MPLS it would again require n labels at each switch, one for every possible destination. In Trainet the number of labels necessary is equal to the number of leaves in the tree (See Lemma 1).

The second idea explored in this paper is to construct a route in the network from several paths each using a different label. This idea may be applied to either MPLS as is or to the Trainet extension. This is much like a subway trip from a station in the Bronx, NYC to a station in Brooklyn, in which several different subway lines going through Manhattan, are required. Notice that the current standard suggested for MPLS supports a pop operation on a sequence of labels in the packet header. This can be used to implement the multi-label route idea in MPLS (assuming that all the necessary LSRs support the pop operation).

The reduction in the number of labels is most important to the size of the ILM table in each label swapping router. Notice that the number of entries in the FEC-Map is not affected by our method since it will always require an entry for each destination or more precisely for each FEC (Forwarding Equivalent Class). However each entry would now contain a <label,counter-value> pair, or a sequence of such pairs. However, the FEC-Map table is not as critical as the ILM table since it is used only once when a new flow enters a Trainet cloud for the first time. The ILM table on the other hand, is used to switch each packet that flows through an LSR.

We consider several possible approaches while implementing these ideas in the following sections.

Topology driven train-lines: This approach which is explored in Section IV uses one <label,counter-value> pair for each route and routes packets on the shortest path from the source to the destination.
Multi label paths in MPLS: In this approach we show how a multi label route in MPLS (without the counters) may save a considerable number of labels. This is explored in Section V.
Multi train-line paths in Trainet - explicit routing: This approach, which is presented in Section VI, considers the usage of multi train-line routes in the Trainet system.
Other possibilities: In all the approaches considered above either a train-line or a multi train-line path, was found to match the route computed by the routing algorithm (be it BGP or OSPF or QoS OSPF). Another possibility which should be explored is to layout an infrastructure of train-lines in the network and then route flows using the existing train-lines, even if the resulting routes do not follow the shortest path. This approach, which is beyond the scope of this paper, raises many questions such as: how to layout an infrastructure of train-lines, how to select a route in such an infrastructure while ensuring good load-balancing and resource sharing.

IV. Topology driven train-lines

Here we consider the implementation of the Trainet method under the following restrictions: (1) that there is a direct train-line from every router (LSR) to any other router, and (2) that this train-line follows the shortest path between these two routers. In this case the train-lines are constructed together with the execution of the routing algorithms (BGP and/or OSPF). It is done by piggy-backing the necessary control and information on the routing algorithm messages. Here, any change in the topology of the network may initiate a change in the train-lines setup.

To understand how this method works consider the shortest path tree rooted at LSR r. This tree is the union of all the shortest paths from r to any other node in the network[1]. The ILM table in r has a train-line for every leaf on the tree and the FEC table would have for every possible destination a pair $< tl,count >$, such that train-line tl goes through this destination and $count$ is the distance from r to the destination.

Claim 1: It is necessary and sufficient to assign a different train-line label for every leaf of the shortest path tree rooted at r, in order to reach every other LSR from r on the shortest path and in one train ride.

Sketch of proof: *Necessary:* Otherwise there are two leaves with the same train-line label. When packets destined to these leaves reach router lca, the least common ancestor of these leaves in the tree, they should continue on different interfaces depending on which leaf is their destination. But if they start with the same train-line then their paths cannot use different outgoing links from lca, a contradiction.

Sufficient: Clearly the train-lines are sufficient for routing packets from r to any other LSR in the network. The fact that this is sufficient also for switching packets from any other source and that pass through r, follows from the optimality principle. To see that, assume by contradiction that there is a source c whose path to destination d, P_{c-d}, goes through r but the segment of P_{c-d} from r to d, P_{r-d}, is not contained in r's shortest path tree (SPT). If the shortest path from r to d has the same length as P_{r-d} then the shortest path from c to d could use this segment instead of P_{r-d}. Therefore assume P_{r-d} is not the same length as the shortest path from r to d. Moreover, by the contradiction assumption the shortest path from r to d is longer than that segment (and hence it cannot use the train-line going out of r and passing at d). A contradiction. The algorithm provided below is another proof of the sufficiency condition. ∎

To implement the principle implied by the above lemma we need an algorithm to construct the ILM table (for switching) and the FEC table (for routing). This routing algorithm is built on the standard BGP like routing algorithm. In such an algorithm every LSR constructs a shortest path tree (SPT) going from itself to every other LSR in the domain. In each step of the algorithm neighboring LSRs exchange their SPTs. Then, based on the SPTs of its neighbors an LSR updates its own shortest path tree. Such an update occurs if it finds a path to a destination shorter than the one in its tree through one of its neighbors trees.

To understand how the above algorithm is adapted to construct the ILM and FEC tables we first focus on how the down stream assignment principle works with the SPTs (much like in Threaded indices [6] and MPLS). Consider an LSR s receiving the SPT from its neighbor t together with a table (ILM) in which t indicates what label it has assigned to each of the leaves in its SPT[2]. The labels s assigns to the leaves of its tree are independent of the labels used by t, however, in the ILM table that s constructs it has to insert an outgoing label for each leaf of its SPT. The outgoing-label that s places in its ILM table for destination d which is a leaf in the SPT of s, and which is reached through t, is an incoming-label that t has assigned to a leaf l of its tree, such that, d is on the path from t to l in t's SPT. See Figure III.

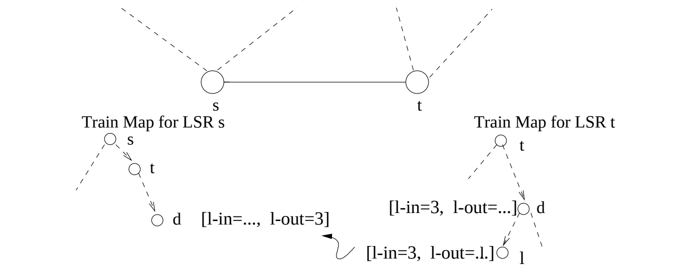


Fig. III. The $l\text{-}out$ that s places for destination d (=3) which is a leaf in the SPT of s, and which is reached through t, is an $l\text{-}in$ that t has assigned to a leaf l of its tree, such that, d is on the path from t to l in t's SPT ($l\text{-}in = 3$ in the example).

The basic principle behind the adaptation of such an algorithm for the construction of the ILM and FEC tables for the Trainet method is as follows: In any iteration in which LSR r constructs a new SPT, different than the one it had before the iteration, it assigns labels to the leaves of the new tree by using as much as possible the labels it had assigned in the previous iteration. The actual implementation of this principle turns out to be full of many small technical details, which are omitted here.

On top of the ILM and FEC tables that the algorithm constructs it uses at each router one more table, the Train-map table which is the SPT of the corresponding LSR and describes all the train-lines going out of a router. This table, which is used only by the algorithm to construct the ILM and FEC tables, is similar in nature and smaller in size, to the BGP table used today.

The main advantage expected from the method is the reduction in the number of labels used by each LSR which equivalently is the reduction in the number of entries in ILM table. To analyze that reduction one has to analyze the number of leaves in the shortest path tree rooted at the LSR in question. Here we suggest two rough estimates on this reduction:

1. The first estimate is based on the relaxing assumption that the SPT is close to a uniform degree balanced tree. Clearly the smaller the degree of the tree the smaller the proportion of the number of leaves in the size of the tree. The average degree in the AS graph of the Internet is about 3.75 [15]. For a uniform tree with branching degree k the ratio between the number of leaves and total number of nodes is $\frac{k-1}{k}$. Thus taking a worse case we may assume an average branching degree in the SPT of $k = 2.75$ ($2.75 - ary$ tree), which results in a ratio of about $\frac{1.75}{2.75} = 63.6\%$ between the number of leaves and the total number of nodes. Meaning the method would save about 36% of the number of labels, or entries in the ILM table. Notice that first the average degree of a SPT in the network of AS's is probably smaller, and second, with the methods described in the following sections we significantly improve the savings in the number labels.

2. In an attempt to obtain a more realistic estimate on the reduction that our method may achieve in the Internet topology we used the data provided in [16] (which was also the basis for the work reported in [15]). First we considered the AS-network, a network in which each AS is a vertex and there is an edge

[1] The algorithm may operate on a network of autonomous systems, i.e., BGP level, or on a network of routers/switches at the OSPF level.

[2] BGP actually sends only differences between the old and new SPT, not complete tables. I.e., it sends only those entries $< dest, AS\text{-}path\text{-}to\text{-}dest >$ in the table which are new.

between two vertices if there is a direct connection between a router in one AS to a router in the second AS. The computed average number of leaves in a shortest path tree in this network is 4099 out of 4746 nodes (different ASs). However, this topology is not relevant since the topology over which BGP is running has many more nodes. The real topology has several BGP-level routers in each AS and these routers are connected both internally and with routers in other ASs. We hope in the future to obtain the topology of this network and then to check the effectiveness of our methods. Moreover, today MPLS is mostly deployed internally, inside ASs networks, and in this networks we believe, following [15], that the ratio between the number of leaves and number of nodes is much smaller. Following [15] the average degree in the network of LSPs inside an AS is 2.57 which means that the ratio of number of leaves in a SPT to the size of the tree is about $\frac{0.6}{1.6} = 37.5\%$. I.e., in these domains we save about 60% of the ILM entries. Again, this is a very rough estimate that we plan to verify.

V. Multi label routes in MPLS

Another idea, independent of the train-line path idea presented in the previous section, and that reduces the number of labels more significantly, is presented here. The idea is simply to specify long routes by more than one label. The general principle is exemplified by considering a long path with a large star network in its end. Under the standard approach, each LSR maintains an LSP for each other LSR, including each of the leaves of the star. However, a simple idea would be to maintain one LSP from each LSR to the center of the star, and then a separate LSP from the center of the star to each of the star leaves. The number of entries in the ILM table in each LSR, except the star center, is now the number of LSRs minus the number of star leaves. This is just an illustrative example below we suggest more general approaches that take advantage of this idea. While this idea is somewhat similar to the usage of virtual paths (VPs) in ATM and the hierarchical routing in MPLS (using the push and pop operations), it is very different from both of these. Both VPs and MPLS push and pop are hierarchical methods while here we combine any two independent paths in the network into one route.

The idea is explored under the assumption that again, topology based shortest path routing is used. The implementation of the idea is based on the stack mechanism of MPLS. That is, a packet that is routed on a two label path carries two labels on its header. First, the packet follows the path specified by the first label, then, the LSR at the end of this path pops the stack of labels carried on the packet. This LSR then routes the packet according to the second label that is now at the top of the packet. Clearly, the idea is not restricted to two labels paths, but here we will mostly consider two labels paths. This idea may considerably reduce the size of the ILM table (notice that the number of entries in the FEC table, which is the routing/forwarding table, can not be reduced by definition). Moreover, some entries in the FEC table now contain the two labels that the route to the corresponding destination follows.

Here we estimate and analyze the amount of reduction in the ILM tables sizes achieved by this idea. We first consider the SPT, and again assume it is a uniform k-way tree of height h. In such a tree we reduce the ILM table size from n, the number of LSRs in the tree to about $\sqrt{n}$ by building an LSP from every LSR to any other LSR that is at most $h/2$ hops away from it. To any other LSR a route would have to be constructed from two LSPs. The number of entries in each ILM follows from the fact that a uniform tree of depth $h/2$ contains about $\sqrt{n}$ nodes (given that a height h tree contains n nodes). In general a uniform k-ary tree of height h has $T(k,h) = \frac{k^{h+1}-1}{k-1}$ nodes in it. The factor by which the number of entries in the ILM table is reduced is $\frac{T(k,h/2)}{T(k,h)}$. For example, for $h = 8$ and $k = 2.75$, n is 5139 (these number are close to the real number of the AS graph induced by the Internet) the number of entries in the ILM is reduced from 5139 to about 89.

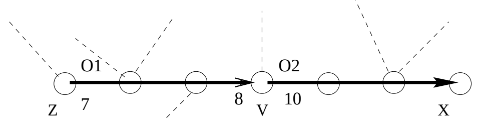
This method can be further improved by combining it with the train-line scheme of the previous section. In such a case every LSR needs to store at most $k^{h/2}$ labels instead of k^h labels. Again in the example above it means storing 57 labels instead of 3270 labels.

Trying to be more realistic we again turn to the paper by Faloutsos et. al. [15] and conclude that the AS graph of the Internet looks as follows: a large and dense transit-ASs core, with many AS stubs hanging off around it. According to [15] about 40% to 50% of the ASs of the Internet are in these stubs (around 1800). Therefore we believe it would be most efficient to establish an LSP between all the AS inside the transit core, and a set of LSPs from every AS to the stub ASs hanging of it. According to the numbers presented by Faloutsos et. al. this alone would reduce the ILM tables size by about half (we have performed similar computations on the data of [16] and arrived at the same conclusion).

Potentially the size of the ILM table may be further reduced by breaking the paths into a larger number of LSPs (or train-lines), which call the multi-label approach. Such a break down in the network of ASs of the Internet would not pay off since the diameter of the network is very small (about 4). However, if we take the whole Internet as a flat network then this method would considerably reduce the ILM table size.

Notice that the way we use the MPLS stack in this section is different than the way it is used in the MPLS standard. There it serves to route packets through a hierarchy such as the AS in the Internet. Upon entering an AS domain, the LSR pushes on the packet header the label that would take the packet on the path to the boarder gate-way on the other end of the AS. When the packet arrives to the boarder LSR on the other side, the stack is popped and the message is routed to the next AS on its route to the destination. Here we use it to achieve some kind of explicit routing, we prepare the stack on the packet header and use only pops to go from one segment of the route to the next. Clearly, the two usages of the stack work together and can be used at the same time.

BGP like routing algorithms can be easily adapted to construct the ILM and FEC tables that support a multi-label or multi-train-line paths. Here we only give the basic idea for the adaptation, the detailed algorithm is beyond the scope of this paper. The rule we assume here is that each path that uses more than c hops should be broken into two paths (we assume no route in the network is longer than $2c$). The routing tables in the routing algorithm would now have the following format at LSR r: for destinations at distance c or less the corresponding entry in the routing table looks as before, $< dest, distance, l\text{-}out, port\text{-}out >$. However, if a destination is at distance larger than c then the entry looks like this (see example in Figure IV): $< dest, distance, l\text{-}out, port\text{-}out, inter\text{-}LSR, inter\text{-}l\text{-}in >$ where, $l\text{-}out$ and $port\text{-}out$ are the label and port leading to the intermediate destination $inter\text{-}LSR$. The LSP that starts by $l\text{-}out$ ends at $inter\text{-}LSR$. When the packet reaches this LSR it is routed according to entry $inter\text{-}l\text{-}in$ in that LSR. In the routing algorithm whenever an LSR receives a table from its neighbor and finds out that its distance to a destination d is more than c it either updates that entry in the routing table with the intermediate information provided by the neighbor, or if the distance is exactly $c + 1$ then it is the first to break the route to d into two. In such a case this LSR updates the entry of d with its neighbor as the intermediate LSR and the label that the neighbor indicated for d as the $inter\text{-}l\text{-}in$. See Figure IV.



Routing table for LSR V

Dest	Dist	l-in	l-out	port-out	inter-LSR	inter-l-in
X	3	8	10	O2		

Routing table for LSR Z

Dest	Dist	l-in	l-out	port-out	inter-LSR	inter-l-in
V	3	11	7	O1		
X	6	23	7	O1	V	8

Fig. IV. The routing tables in the routing algorithm that constructs multi train-line routes. In order to reach X, Z pushes multi label - $\mathbf{7}|\mathbf{8}$: The head of the stack $\mathbf{7}$ - which is the label in order to reach V and under it $\mathbf{8}$, which is the label that V assigned for reaching X.

The decision where to break a route into two or more pieces can be either by a distance parameter as described above or according to specific end points such as core boarder LSRs etc.

VI. Multi train-line routes and explicit routing

One of the important features of MPLS is its ability to enforce load-balancing and traffic engineering. This feature is achieved in MPLS by establishing an explicit route in the network to which a label is assigned. Using the label that was assigned to the explicit path supports fast switching, and small headers. However, the assignment of a unique label to each explicit path in the network requires a large number of labels, which increases the ILM table sizes. This places an extra demand on the already scarce resource, the labels. In IP, on the other hand, explicit routing is implemented by each packet carrying the entire list of all routers along the explicit route. (Alternatively, one can use tunneling in IP, but this feature still uses the IP routing and forwarding, i.e., no reduction in tables sizes and no fast switching on a label). Thus the packets traversing the explicit route in IP carry a large header and require more processing at each router along the way.

Constructing an explicit route from multi train-lines presents a promising trade-off between the MPLS explicit routing and the IP explicit routing. The multi-train explicit routing enjoys from the advantages of each. It achieves both small packet headers, and fast label based switching, while we do not impose an increase in the number of labels in the network. The path is now specified by a list of the relevant train-lines and the number of hops to take with each (see Figure V). In each the packet is forwarded by switching on the label where in intermediate stations a label popping is performed (see below how penultimate hop popping [17] may be applied to make this operation as efficient as label swapping).

The tradeoff between the number of labels necessary, and the length of the train-lines list, is an issue for further optimization and research. Longer lists increase the bandwidth consumed by the headers but reduces the number of labels required in the network, thus keeping the ILM tables small. The general idea is to enforce a maximum on the number of train-lines per explicit route. Whenever a route traverses more than that number of train-lines, a new train-line is established along this route.

For the rest of this section we assume the following framework; At any point in time some collection of train-lines is already laid out in the network with each LSR having information on the trains that pass though it (we consider different variation in which the amount of information each LSR has on the train-lines going through it is different). When a new explicit connection is being setup, there is a routing mechanism that selects the route for the new connection in the network. This routing mechanism is outside the scope of this section, here we only assume that the route selected is given to the explicit routing setup mechanism. Below we consider different interfaces between the route selection process and the setup process (i.e., hop by hop, or complete route in one shot). From time to time it is possible that new train-lines are defined in the network, either because some part of a route has no train-line covering it, or because some route is stitched together out of too many train-lines and we would like no route to make more than a certain number of train-lines exchanges. In the rest of this section we address the following issues: how the setup of an explicit route sets up the route, how to add new train-lines, and how this mechanism may be combined with RSVP-TE (or CR-LDP).

Packet forwarding along an explicit route of train-lines is carried out by label swapping. Upon receiving a packet each LSR on the route decrements the hop-count at the top of the stack on the packet. If the counter value is still greater than zero the packet is forwarded on according to the label at the top of the packet stack. On the other hand, if the resulting value is zero then it pops the stack and performs another switching operation (looking up in the ILM table) on the new top of the stack. Notice that in this last case, the LSR performs two look up operations at the head of the stack, and two look up operations in the ILM table.

Here we can use in a natural way the Penultimate Hop Popping mechanism described in [17]. The LSR before the one that has to perform a pop operation (i.e., the one that decrements the counter from two to one), pops the stack and sends the packet to the next LSR. By this we accomplish that the LSR and the Penultimate LSR, both perform only one look up at the ILM table and one lookup at the head of the stack.

We distinguish between two separate functions in explicit routing, the route selection function, and the explicit route setup. The route selection is implemented by some mechanism, such as QoS routing, or some other routing mechanism. The setup function receives from the route selection function the directions, where to go and what are the LSRs on the route. The setup function establishes the explicit route which in our case corresponds to the selection of a set of train-lines, each with its corresponding number of hops. In what follows we deal with the explicit route setup given that some infrastructure of train-lines is present, and that the route selection procedure is given.

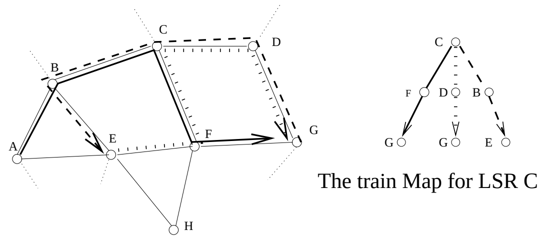


Fig. V. Multi train-line: The route A-B-C-D-G is defined by the sequence of train-lines $< x, 2 >, < y, 2 >$, where x is the label at LSR A for the train-line with the solid line, and y is the label at LSR C for the train with the dotted line.

A. *Train-line by train-line setup*

We suggest two approaches to the explicit route setup. I.e., to the procedure by which the set of train-lines that covers the route is determined.

In the first approach we assume that each LSR maintains a train-map data structure which lists all the train-lines going out of this LSR with their detailed routes, and that for each new flow the route that the flow should take is fully specified. This train-map data structure is the same train-map that was used in the routing algorithm briefly described in the previous section. The train-map data structure is most efficiently represented by a trie (kind of tree). Each path in the trie represents a train-line. If a prefix of a train-line is also a prefix of other train-lines the it is represented only once in the trie. Notice, that since we don't restrict ourselves to shortest path, the same LSR might appear more than once in the trie (in case it is reached with two trains that go on different paths).

The basic principle here is that s, the first LSR selects the train-line that takes us the most number of hops along the explicit route among all the train-lines going out of s. Then, this train-line and number of hops is recorded on the setup message which is then sent on that train-line to the last point of the train-line which is on the explicit route. At this point the procedure reiterates, selecting again the train-line from all the train-lines going out of the new point, that makes the maximum number of hops along the rest of the explicit route. This process is continued until the end of the explicit route is reached. At that point the set of train-lines with the respective number of hops is recorded on the setup message that is now sent back to the source.

Notice that for this method to work there must be a train-line going over each link along the selected route. If this is not the case, the setup message is sent over the link and when the message is sent back from the destination of the route to its source a train-line is extended over this link, or a new train-line is defined over the link. The details of this procedure are left for the full paper.

Claim 2: The algorithm described above finds the smallest number of train-lines that cover a given explicit route.
See Appendix A for the proof. ▮

In the second approach we assume that a train-map is not available at the LSRs. In this case, each LSR has an FEC table and an ILM table. I.e., for each outgoing link an LSR knows what set of trains goes out on this link. The setup starts by the first LSR placing on the setup message the set of all the train-lines that go over the first link of the explicit route. Each LSR down the explicit route checks which of the set of train-lines recorded on the setup message continues over the next link of the explicit route. It then leaves on the setup message only this subset of train-lines and sends the message down the explicit route. This process repeats until reaching an LSR in which none of the train-lines on the setup message continues on the next link of the explicit route. At this point one of train-lines that were still recorded on the setup message is selected as part of the explicit route and the current LSR starts the process from the beginning, placing the set of all the train-lines going out of it on the next link on the setup message and so forth.

Notice that the last approach can be applied in case the route selection is done on a hop-by-hop basis. I.e., the setup would be integrated with the route selection protocol. Furthermore, the setup can be done also on the way back from the receiving end point of the explicit path. However we postpone this discussion to the full version of the paper.

In the first method the setup messages are somewhat smaller (no need to send a list of train-lines), but each LSR is required to hold a train-map which might be a large data-structure. In the second method, on the other hand, the setup messages might be larger since they carry a set of train-lines, but the LSRs do not have to maintain any extra data-structure. It is possible to combine the two approaches above into a hybrid approach. The hybrid uses a small train-map table up to a certain distance at each LSR, and requires a smaller setup message.

B. *On demand establishment of a new train-line*

In this subsection we consider the case in which it is necessary to establish a new train-line in the network. This happens in either one of the following cases: (1) we find no train-lines that goes over a route, or part of the route over which we need to set an explicit route, or (2) the minimum number of train-lines that cover an explicit route is more than the maximum number of train-lines we allow for one explicit route. Composing a route out of too many train-lines is problematic for two reasons. First, the size of the header of messages that use this explicit route would be proportional to the number of train-lines used, i.e., it would imply a large header. Second, the overhead in switching the packets along this explicit route would be large because on many LSRs we would need to do a pop and some minimal processing of the packet header.

When a new train-line is established over a particular path in the network there are several optimizations that we should pay

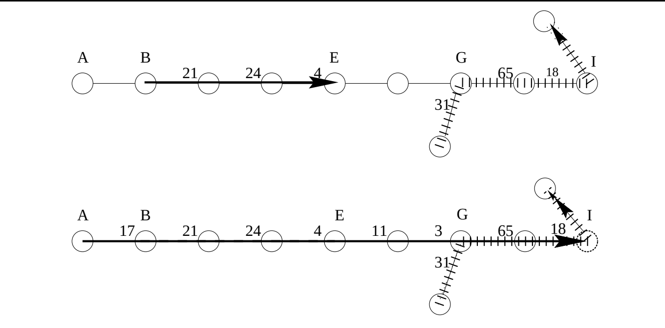


Fig. VI. The lower half describes a network after a train between A and I was established. The upper half describes the situation before the establishment of the new train-line. The number on every edge, is the incoming label to the next LSR in the direction of the train-line. Notice how the train-line between A and I, is using the label that was bound to the train-line between B and E and part of the train-line that was between G and I. And by this eliminating a need of more entries in the ILM in the LSR of the new train.

attention to. These may save the usage of new labels in some of the LSRs along that path. The basic principle behind these optimizations is the following (see Figure VI): When a new train-line is formed over a path, and part of the path is already covered by an old train-line such that the final LSR of the old train-line is on this path, then, any LSR that is covered by the old train-line does not need any new label for the new train-line. Any such LSR may use the label it used for the old train-line in the formation of the new train-line.

C. Explicit Route binding protocols with multi train-lines

In this subsection we describe how the multi-train-line explicit route setup may be integrated into the RSVP-TE and/or the CR-LDP protocols. In MPLS explicit routes may be established by the RSVP-TE protocol [18] or the CR-LDP [19].

We describe here how the train method can be integrated with RSVP-TE (the integration with the CR-LDP is similar). In RSVP-TE the explicit route establishment is piggy-backing on the RSVP protocol. The PATH forward message of RSVP, travels according to the explicit route that is saved in the Explicit Route Object (ERO). When the PATH message reaches the destination, the receiver initiates a RESV message in the backwards direction. On this message the receiver declares the reservations it needs. RSVP-TE uses this RESV message to do the downstream label binding by piggy-backing the labels on the RESV message.

The setting up of a multi-train-line explicit route via the RSVP-TE protocol is done as follows: The RSVP PATH message (that travels the path in the forward direction) is used to collect the sequence of train-lines that compose the route, as described in Section VI-B. When this message reaches the receiver end-point of the path, that LSR decides whether a new train-line should be formed along the route (because either some links along the path had no train-line going over them, or because too many train-lines were necessary in order to cover the route). If a new train-line is necessary then the RESV message that goes backwards would setup the new train on its way back, or if a new train-line is not necessary the RESV message would simply carry the sequence of train-lines that compose the new explicit route (as was recorded on the PATH message). Notice that establishing a new train-line on a message that goes backwards on the path of the new train-line is rather easy. It is the direction necessary for downstream assignment, and at each LSR on the way it may update the train-map with the new train.

A similar procedure may be implemented in CR-LDP. There the Label Request message plays the role of the PATH message, and the Label Mapping plays the role of the RESV message.

VII. Multi train-lines and fault-tolerant bypassing

The main disadvantage of label switched paths or any other end-to-end circuit switching is the slow recovery from link or router failures. The reason being the complex and time consuming process of setting up a new LSP. Especially when the new LSP selection is based on specific traffic engineering requirements [20]. This stands in contrast to the ease in which IP and other hop by hop routing methods seamlessly adapt to failures in the network.

The multi label train-line idea of Section VI, provides us with mechanisms by which an LSP may be quickly mended and patched. Basically, an LSR on the path that detects the failure of the next link or router, locally computes a detour and inserts a sequence of train-lines that covers the detour into the multi label train-lines that compose the path.

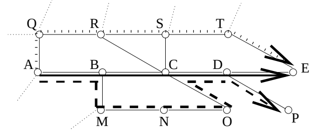


Fig. VII. Bypassing example, see explanation in the text.

Consider for example the scenario in figure VII, in which there is a train-line, r, going over $A - B - C - D - E$, and another train-line, called z going over $A - B - M - N - O - C - D - P$ (For clarity reasons r and z are each a single train-line however each may be composed of a sequence of train-lines).

If link $(B - C)$ fails then B may perform a fast route recovery by inserting 4 hops of train z (i.e., $M - N - O - C$) and then continuing from C with train r. In the process of finding the detour B needs to check if the detour can hold the traffic requirements of the original train. The information about the detour links may be retrieved from the train map. Notice that B needs to know the label describing train r in C. This information may be obtained during the process of building the train-map.

A similar idea was presented in [21] where a bypassing technique for source routing and ATM networks is described. Since the LSR that chooses the detour may do so by looking only in its local data, the selection is fast.

The disadvantage of this method, is that since the detour is chosen locally it may not be the optimum detour, that would have been chosen if the source LSR of the flow would have calculated a new path in a global way. In our example A could use a train-line that goes through the path $(A - Q - R - S - T - E)$.

VIII. Future work

We believe the present paper leaves many open questions at different levels of abstraction. The main issue is what is a “good” and useful extension of MPLS. The extensions suggested here need to be studied in further detail to prove (or disprove) their effectiveness. We believe that either the multi-label idea or the train-line idea, each separately may result in a considerable reduction in the ILM tables, i.e., in the number of labels required in the network. This may prove to be an important step in the scalability and efficiency of the label swapping approach. The first step in this study would be to check the implementation of the ideas on real samples of the Internet topology. This is a non-trivial and major task. Measurement of the number of labels required by MPLS and IP-switching were performed in [22], [23].

In all the variations considered in this paper the routes selected were assumed to be selected by some routing mechanism, either shortest path (like BGP or OSPF) or some other. There are different approaches that should perhaps be looked at. One is to layout some set of train-lines in the network, and then to perform routing based on the available set of train-lines. This approach looks problematic because first, it does not select shortest paths, i.e., it may consume more of the network resources then it has to. Second, it requires a new routing protocol. Furthermore, this method raises other issues such as: what initial infrastructure of train-lines should we layout. Should it be done on demand? or by predicting hot spots, bottleneck links and make them part of a few train-lines.

In another approach we would relax the requirement to use shortest paths and study the same set of questions as above but allowing a stretch factor, i.e., the routes may be longer by a factor ϵ than the shortest routes. Another question is to study the tradeoff between the ILM table size and the number of labels (or train-lines in Trainet) necessary to compose any route in the network (in either case of shortest paths, or allowing a stretch factor).

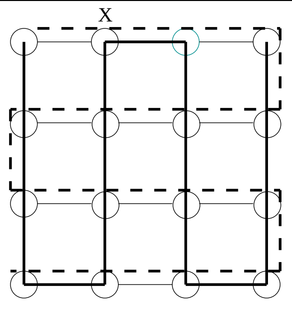


Fig. VIII. The Grid Network. Two bi-directional trains: one bi-directional train in solid line, and another in dotted line, are sufficient to compose all the shortest paths on the grid.

Consider for example the following grid in figure VIII. In this topology we can construct any shortest path in the network by combining two train-lines and setting only four trains in the whole network. The main idea is to set up one two directional train-line (i.e., two train-lines, one for each direction) that passes through all the rows, and other two directional train-line that passes through all the columns. Hence in order to get to a LSR, a multi train-lines contains one line to get to the LSR destination raw, if needed, and another line to get to the LSR destination column, if necessary. Similar routing protocols for grid networks where suggested in the Manhattan street network [24] and by Myrinet [25].

Acknowledgments: We would like to thank Sariel Har-Peled for many helpful discussions.

References

[1] B. Davie, P. Doolan, and Y. Rekhter, *Switching in IP Networks*. Morgan Kaufmann Publishers Inc., 1998.
[2] Y. Rekhter, B. Davie, D. Katz, E. Rosen, G. Swallow, and D. Farinacci, “Tag switching architecture overview,” 1996. ftp://ds.internic.net/internet-drafts/draft-rfced-info-rekhter-00.txt. Work in progress.
[3] “Aris website.” http://www.networking.ibm.com/nsw/aris.html.
[4] Y. Katsube, K. Nagami, and H. Esaki, “Toshiba’s router architecture extensions for atm: Overview. rfc 2098,” tech. rep., April 1997.
[5] P. Newan, G. Minshall, and L. Huston, “Ip switching and gigabit routers,” *IEEE Communications Magazine*, Junary 1997.
[6] G. Chandranmenon and G. Varghese, “Trading packet headers for packet processing,” *IEEE Transactions on Networking*, April 1996.
[7] R. Callon, P. Doolan, N.Feldman, A. Fredette, and G.Swallow, “A framwork for multiprotocol label switching,” tech. rep., IETF, November 1997. draft-ietf-mpls-framework-02.txt. Work in progress.
[8] M. Waldvogel, G. Varghese, J. Turener, and B. Plattner, “Scalable high speed ip routing lookups,” in *Proc. ACM SIGCOMM 97*, Octeber 1997.
[9] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, “Small forwarding table for fast routing lookups,” in *Proc. ACM SIGCOMM 97*, Octeber 1997.
[10] B. Lampson, V. Srinivasan, and G. Varghese, “Ip lookups using multiway and multicolumm search,” in *Proc. INFOCOM 98*, March 1998.
[11] S. Nilsson and G. Karlsson, “Fast address look-up for internet routers,” in *Proc. IEEE Broadband Communications 98*, April 1998.
[12] N. F. Huang, S. M. Zhao, J. Y. Pan, and C. A. Su, “A fast ip routing lookup scheme for gigabit switching routers,” in *Proc. INFOCOM 99*, March 1999.
[13] P. Gupta, S.Lin, and N. McKeown, “Routing lookups in hardware at memory access speeds,” in *Proc. INFOCOM*, April 1998.
[14] T. C. Chiueh and P. Pradhan, “High performance ip routing table lookup using cpu caching,” in *Proc. INFOCOM 99*, March 1999.
[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On power-law relationships of the internet topology,” in *Proc. ACM SIGCOMM 99*, Sept 1999.
[16] “Network analyis infrastructure.” http://moat.nlanr.net/.
[17] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol label switching architecture,” 1999. draft-ietf-mpls-arch-05.txt. Work in progress.
[18] D. Awduche, L. Berger, D. Gan, J. Networks, T. Li, G. Swallow, and V. Srinivasan, “Extensions to rsvp for lsp tunnels,” March 1999. draft-ietf-mpls-rsvp-lsp-tunnel-02.txt. Work in progress.
[19] B. Jamoussi, “Constraint-based lsp setup using ldp,” February 1999. draft-ietf-mpls-cr-ldp-01.txt. Work in progress.
[20] B. J. L. Andersson, B. Cain, “Requirement framework for fast re-route with mpls,” Oct 1999. http://shika.aist-nara.ac.jp/member/nori-d/mlr/id/draft-andersson-reroute-frmwrk-00.txt. Work in progress.
[21] R. R. I. Cidon and Y. Shavitt, “A fast bypass algorithm for high-speed networks,” in *Proc. INFOCOM 95*, April 1995.
[22] H. Esaki, “Evaluation of flow aggregated traffic driven label mapping policy in mpls system,” tech. rep., UUNET IW-MPLS, 98. http://info.uu.net/ads/techconf/presentations/.
[23] S. Lin and N. McKeown, “A simulation study of ip switching,” in *Proc. ACM SIGCOMM 97*, September 1997.
[24] N. Maxemchuk, “Routing in the manhattan street network,” *Transactions on Communications*, vol. 5, pp. 503–512, May 1987 1987.
[25] A. D. R. Felderman, D. Cohen, and G. Ginn, “Atomic: A high speed local communication architecture,” *Journal of High Speed Networks*, vol. 3, no. 1, pp. 1–30, 1994.

Appendix

I. Proof of Claim 2

Claim 2 The algorithm described in Section VI-A finds the smallest number of train-lines that cover a given explicit route.

Proof: Assume to the contrary: There is an explicit route $LSR_1, LSR_2, ..., LSR_n$ that, the algorithm finds for the above route the multi train-lines path: $< tline_1, n_1 >, < tline_2, n_2 >, \ldots, < tline_l, n_l >$. But there is another multi train-lines path: $< tline^*_1, n^*_1 >, < tline^*_2, n^*_2 >, \ldots, < tline^*_m, n^*_m >$, covering the same route, such that, $m < l$.

We denote by k_i the number of LSRs that were traversed after using the first i train-lines (e.g., the $tline_{i+1}$ is taken from LSR_{k_i}). Formally, $k_i = \Sigma^i_{j=1} n_j$. Similarly we define k^*_i.

By induction we prove below that for each i, $1 \leq i \leq m$, $k_i \geq k^*_i$. Hence this holds for $i = m$ and $k_m \geq k^*_m$. But by the assumption $k^*_m = n$ and $k_m < n$. A contradiction.

We now prove the inductive assumption. For the base case, $i = 1$: LSR_1 chooses the train-line that takes the most number of hops along the explicit route. Hence, $n_1 \geq n^*_1$, and $k_1 \geq k^*_1$.

Assume that the inductive assumption, $k_j \geq k^*_j$ holds for $j = 1, \ldots, i$ we prove that this assumption holds for step $i + 1$. Assume the opposite, the assumption does not hold for $i + 1$: i.e., $k_{i+1} < k^*_{i+1}$. We get, $k^*_i \leq k_i < k^*_{i+1}$ (since by inductive assumption $k^*_i \leq k_i$, and by definition $k_{i+1} = k_i + n_i$ and we assume by the contradiction assumption that $k_{i+1} < k*_{i+1}$).

Hence the train-line $train\text{-}line*_{i+1}$ passes through LSR_{k_i}. By the algorithm the train Map of LSR_{k_i} contains this train-line, and since the algorithm chooses the train-line that takes the most number of hops along the path we get that $k_{i+1} \geq k*_{i+1}$. Contradiction. ▮