Available online at www.sciencedirect.com
SCIENCE DIRECT®
Microprocessors and Microsystems 28 (2004) 439–445
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Loop detection in MPLS for wireless sensor networks☆

Vasu Jolly[a], Naoto Kimura[a], Shahram Latifi[a], Pradip K Srimani[b,*]

[a]Department of Electrical Engineering, University of Nevada, Las Vegas, NV, USA
[b]Department of Computer Science, University of Clemson, 401 Edwards, Clemson, SC 29634, USA

Received 28 August 2003; revised 26 February 2004; accepted 1 March 2004

Available online 26 March 2004

Abstract

Very few technologies illustrate the fast rate of technological innovations more than wireless sensor networks. Sensor networks offer a virtual path capability to carry differentiated services efficiently across the wireless backbone. In this paper, we provide a new efficient strategy for loop detection in Multi-protocol Label Switching (MPLS) for wireless networks—MPLS is a novel wireless networking topology that can be used to provide differentiated service, traffic engineering and quality of service in wireless networks.

Keywords: Sensor networks; Power aware protocols; DPM; SPM; Loop detection; Multi-protocol label switching

1. Loops in wireless sensor networks and MPLS

The astronomical growth of the Internet communication offers a serious challenge for network planners in terms of heavy user traffic. The Internet core is continuously being expanded to meet growing bandwidth demands. The growth in bandwidth demand hinders core providers' ability to add infrastructure. Besides issues of resource constraints, another concern is to significantly transport bytes over the backbone to provide an efficient class of service for the diverse requirements of the users, such as multimedia applications. To manage the above addressed issue, there is a need to either increase the bandwidth of existing circuits or the capacity of the core routers apart from adding more core routers. In general, network providers need to be concerned about scalability issues, which can escalate the ability to expand the network in all the dimensions. Multi-protocol Label Switching (MPLS) based on Label switching offers an ability to build highly scalable networks. The greatest strength of MPLS is its coexistence with IP traffic and its reuse of IP routing protocols. It encapsulates the dexterity of routing with the performance of switching providing relevance to networks with a pure IP architecture as well as those with IP and ATM or combination of other Layer 2 technologies.

MPLS [1,3,4,6] is rapidly emerging as an Internet Engineering Task Force (IETF), standard intended to enhance the speed, scalability and service providing capabilities in the Internet. MPLS uses the technique of packet forwarding based on labels, to enable the implementation of a simpler high-performance packet-forwarding engine. This also de-couples packet forwarding from routing, facilitating to provide varied routing services independent of the packet forwarding paradigm. The evolution of this technology in relation to other existing technologies is tracked. In MPLS, a small fixed format label is encapsulated within each data packet on its entry into the MPLS network. In router networks, the label is a separate, 32-bit header. In ATM networks, the label is placed into the Virtual Path Identifier/Virtual Channel Identifier cell header [7–10].

In the MPLS core, Label switched Routers (LSRs) read only the label, not the network layer packet header. Labels have only local significance between two devices that are involved in communication. At each hop across the network, the routing of the data packet is based on the value of the incoming label and eventually issued to an outwards interface with a new label value. The path that data traverses through a network is defined by the transition in label values, as the label is swapped at each LSR. Since the mapping between labels is constant at each LSR, the path is determined by the initial label value. Such a path is called

☆ Last authors work was partially supported by an NSF award ANI-0218495.
* Corresponding author. Tel.: +1-864-656-7552; fax: +1-864-656-0145.
E-mail address: srimani@cs.clemson.edu (P.K. Srimani).

0141-9331/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2004.03.011

a Label Switched Path (LSP). At the ingress to an MPLS network, each packet is examined to determine which LSP it should use and hence what label to assign to it. Here, the IP packets are classified based on the information carried in the IP header of the packets and the local routing information maintained by the LSR and a label is assigned to them. The labels are then distributed to the neighboring LSRs, and further associates and distributes till the egress LSR is reached. Each LSR uses the label to forward the packet. At each LSR the outgoing label replaces the incoming label and the data packet is switched to the next LSR. The process of switching the label is known as Label Swapping. The set of all packets that are forwarded in the same way is known as a Forwarding Equivalence Class (FEC). One or more FECs may be mapped to a single LSP. Classification and filtering of the information packet happen only once, at the ingress edge. At the egress edge (output routers), labels are stripped and packets are forwarded to their final destination.

Fig. 1 depicts two data flows from workstation 2 to workstation 5. LSP is shown connecting LER1 and LER 2. LER 1 is the ingress point into the MPLS network for data from workstations 1–3, respectively. A packet enters the ingress Edge LSR (LER 1) where it is processed to determine which Layer 3 services it requires, such as QoS and bandwidth management. Based on routing and policy requirements, the Edge LSR selects and applies a label to the packet header and forwards the packet. Thus, LER 1 determines the FEC for each packet, deduces the LSP to use and adds a label to the packet. LER 1 then forwards the packet on the appropriate interface for the LSP.

LSR 1 is an intermediate LSR in the MPLS network. It simply takes each labeled packet it receives and reads the label on each packet, replaces it with a new one as listed in the table, uses the pairing {incoming interface, label value} to decide the pairing {outgoing interface, label value} with which to forward the packet and finally forwards the packet. This action is repeated at all LSRs, till the time it reaches LER 2.The incoming label and corresponding outgoing labels are stored in a table, known as the forwarding table. The swapping of label value and forwarding of the packet can be performed in hardware. This allows MPLS networks to be built on existing label switching hardware such as ATM and Frame Relay. LER 2 acts as egress LSRs from the MPLS network. These LSRs perform the same lookup as the intermediate LSRs, but the {outgoing interface, label value} pair marks the packet as exiting the LSP. The egress Edge LSR (LER 2) strips the label, reads the packet header, and forwards it to its final destination using layer 3 routing. So, if LER 1 identifies all packets for ws-5 and appropriately labels them they will be successfully forwarded through the network. In MPLS, data transmission occurs on label-switched paths (LSPs). LSPs are a sequence of labels at each and every node along the path from the source to the destination. LSPs are established either prior to data transmission (control-driven) or upon detection of a certain flow of data (data-driven). The labels, which are underlying protocol-specific identifiers, are distributed using label distribution protocol or RSVP or piggybacked on routing protocols like border gateway protocol and OSPF. Each data packet encapsulates and carries the labels during their journey from source to destination. High-speed switching of data is possible because the fixed-length labels are inserted at the very beginning of the packet or cell and can be used by hardware to switch packets quickly between links.

The issue of transient loops for large router networks is currently addressed with utmost importance in MPLS environment. The asynchronous behavior of LSRs and Link failures in the chain of routers or hubs, sometimes causes control path to jump into an oblivious loop behavior, which results in an establishment of a LSP along the routing loop, until it breaks of itself. Control packet needs to be discarded once this behavior is detected or halted to be re-routed from an alternate path. In MPLS scenario, since the labels are distributed and the path for the data packets is set

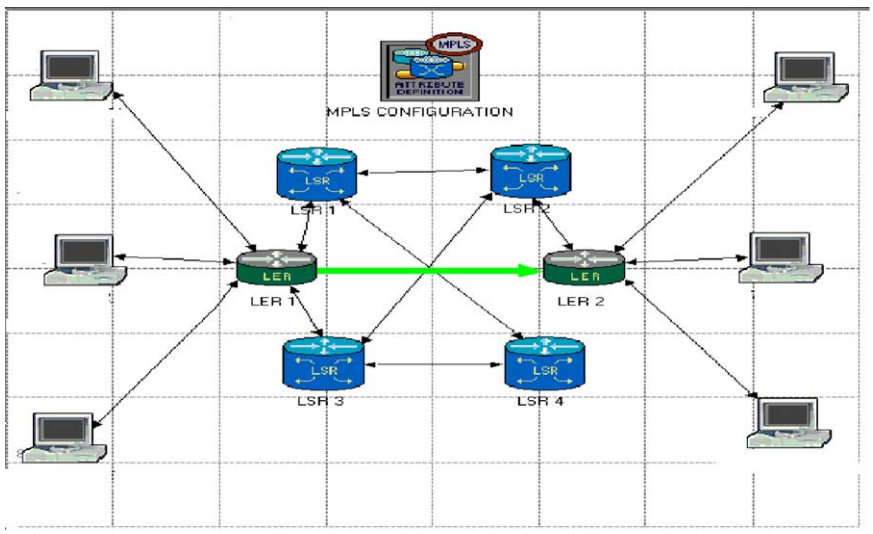

Fig. 1. Workstations communicating under MPLS environment.

beforehand, the loop formation occurs at the control path. Discussion of these issues forms a large part of the Framework document at MPLS Work Group. Ohba [2] addressed the problem of Loop Detection and stressed the need to eradicate loop formation in MPLS networks. Pertinent information about forwarding data packets needs to be established by each node, within a network. A network performance can remarkably degrade due to existence of an undesirable loop. The occurrence of loop formation in MPLS is generally a less frequent phenomenon, but needs to be dealt with a higher order of precision to avoid abrupt data losses. The loop avoidance mechanism should not be too complex to devour the router's computational power by gulping the router's memory. Rather, it should be simple and effective.

Currently two loop prevention algorithms have been proposed to the IETF [2], which is path-vector/diffusion algorithm and colored thread algorithm. The mechanism for the loop detection and prevention establishes running a thread hop-by-hop before the labels are distributed inside a MPLS cloud. With the passage of the each next hop, a distributed procedure is executed within the thread mechanism. The present work in this paper is a brief overview of the existing loop prevention mechanism, besides using the global variables, instead of IP addresses for comparative smaller network cloud. The existing loop prevention scheme is briefed which ensures loop detection and loop mitigation Furthermore, a suggestion of assigning the labels, while rewinding the thread has been given, which could substantially reduce the LSP set up time and add to the efficient thread mechanism.

2. Loop formation in MPLS

The loop formation within the nodes or routers is an unfavorable phenomenon. With the flow of data packets, each node needs to be updated and synchronized, according to any of the existing routing algorithm, such as shortest path between nodes or less congested path backbone. The inconsistency in refreshing the routing information causes loops to get formed and data packets to move within the loop without reaching the destination. If loop formation is not controlled, it leads to control packet looping, where packets used for establishing a LSP continue to be forwarded along the routing loop until the routing loop breaks either by itself or explicitly. Fig. 2 shows a network with multiple paths existing from a source (S) to a destination (D) at any given time. We use a shortest Path algorithm, considering the distance from LSR A (source) to LSR I (destination). In MPLS, a control path is generated before the actual data can be transmitted. In this control path, the task of label assignment and label distribution is accomplished. Considering the output of this algorithm, the path A–B–E–F–I from LSR A to LSR I is the shortest one. This particular structure can be extended to any generalized

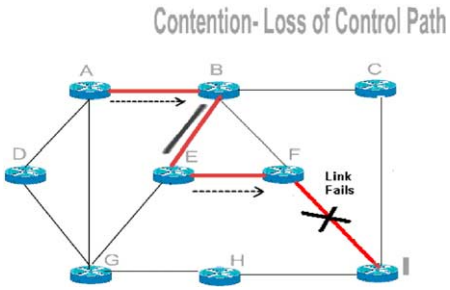

Fig. 2. Simple loop formation.

case in the cluster of networks, as the weight between any two nodes is the major factor in resolving the actual shortest path. At this point, it should be noted that the data flow has not yet taken place; it is just the label assignment, which gets initiated. Theoretically, all the router nodes should get refreshed simultaneously and in synchronization with real time. Assuming that a link between LSR-F and LSR-I fails, some data packets destined for LSR-I have already departed from node A to F. Node F would have to send back the control packets and has to reroute it from a different path, which should be the shortest of all available paths. Now LSR-F takes another short path: F–E–B–A–G–H–I. However, LSR-B may still stick to the previous shortest path, without knowing the failure between LSR-F and LSR-I. In this case, LSR-B continues to send control packets towards LSR-E and LSR-F. On the other hand, LSR E tends to send the control packets towards LSR-B. Thus a loop gets formed between LSR-B and LSR-E, resulting in a loss of the control path. Though this loop occurrence is rare and transient, it has to be removed for an efficient set-up of a label path and later the data path, which results in an efficient flow of data. Furthermore, without any loop avoidance algorithm installed, it should be noted that as the loop gets larger and more complex, it takes more time for the system to trace it manually and to come out of the loop. Fig. 3 reveals a complex loop formation for the previous network, which has the shortest control path from LSR-A (S) to LSR-I (D): A–B–E–F–I.

If a link between LSR-F and LSR-I fails, some of the data packets destined for LSR-I have already departed from node A to F. In this case, the LSR's A, B, E and F have been refreshed and understand the failure of link F-I. The rerouting takes place from F–E–B–A–G–H–I.

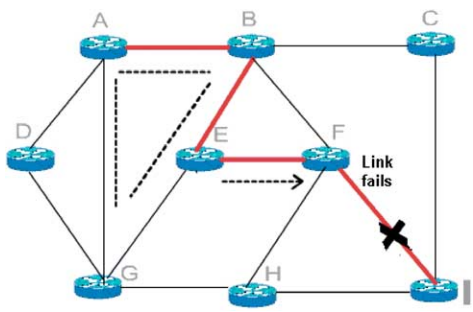

Fig. 3. Complex loop formation.

Furthermore, we assume that LSR G is not refreshed at this time, and in its information database {LSR-G's}, the most optimal path from G–I is G–E–F–I. As soon as the rerouting takes place, the control path follows F–E–B–A–G and towards E–F–I, which results in the formation of a complex loop G–E–B–A.

2.1. Colored thread algorithm

RFC 3063 [5] addressing Colored Thread algorithm is categorized as an experimental standard and currently is the part of research and experimental effort. In this section, a mechanism for generating a thread is explained and the basic thread actions are explained [5,6]. The examples showing the launch and the end of threads have been explained in the following subsections. The importance of thread mechanism is addressed and its relevance to the current loop prevention scheme is discussed in detail.

2.2. Thread attributes

A thread is a sequence of messages used to set up an LSP, in the 'ordered downstream-on-demand' (ingress-initiated ordered control) style. There are three attributes related to threads. They may be encoded into a single thread object as.

2.3. Thread color

The sole purpose of assigning a color to respective threads is to assign a unique entity to the path control message. Since the color has to be unique in time and space, thus ensuring the interface between the LSRs to be unique. When the thread is allowed to pass through LSRs, these unique colors will be assigned to each interface and the results be stored and maintained by the nodes. It should be noted that a thread be called transparent, when all the fields in it, are zeroes and is reserved for stalling of thread.

COLOR = IP ADDRESS + UNIQUE IDENTIFIER

A 16 bit unique number is selected on the random basis, and is allowed to be incremented by a fixed interval, thus by enabling color to be unique and ensuring that while working with independent nodes, the same color does not get repeated. In this method, the initial event identifier is either selected at random or assigned to be larger than the largest event identifier used on the previous system incarnation.

2.4. Thread TTL

A Time to Live (TTL) field is added to a thread whenever a node creates a path control message This TTL field, decreases with one bit of each hop. To prevent the unnecessary looping actions in a network, the message should not be forwarded when, TTL reaches 0. The TTL is set by the sender to the maximum time the thread is allowed to be in the network. If the thread is in the Internet system longer than the TTL, then the thread must be destroyed. The field must be decremented by one. The time is measured in units of seconds (i.e. the value one means 1 s). Thus, the maximum TTL is 255 s or 4.25 min (Fig. 4).

2.5. Thread hop count

Thread hop count is the field, which starts from a minimum value (say 1), from the ingress node, and keeps on increasing uniformly (by one), with each hop change When the ingress node assigns a hop count of one to its downstream link, it stores this value and jump to the next LSR, and it happens for all the LSRs in the network. When a loop is found, a special hop count value = (0XFF) is assigned, which should be larger than 256 (corresponding decrementing TTL value). When the same colored thread is received on multiple incoming links, or the same thread

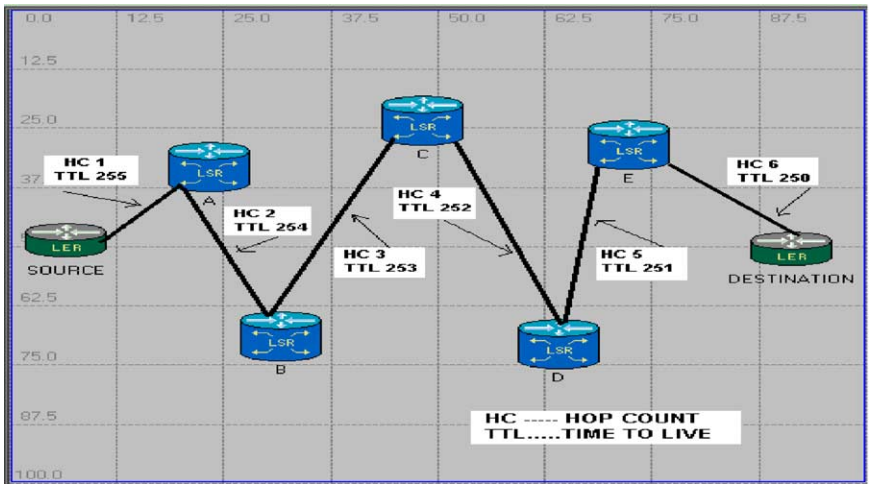

Fig. 4. Hop count and time to live.

color was assigned to the node again, it is said that the thread forms a loop. A network manager can judge whether it assigned the received thread color by checking the IP address part of the received thread color.3.5. A thread is said to form a loop when the thread of same color is received on an incoming link of the router. A thread creator can detect it, by checking the IP address field of the LSR. The basic thread actions to prevent LSP loops include 'thread extending', 'thread rewinding', 'thread withdrawing', 'thread merging', and 'thread stalling'.

2.6. Thread extending

Extension of thread plays a pivotal role in color thread algorithm [2] Before setting up a LSP and assigning the respective labels to each LSR, a thread, needs to be extended from the source node to the destination node. The thread creation starts from the ingress node and ends at the egress node. Each respective node from source until destination creates a thread, assigns color and extends it downstream. The color and the hop count of each thread, becomes the color and hop count of the outgoing link. In other words, for the ingress node, the hop count is set to be one; the TTL field is set to be its maximum value, 256. The color assigned to the thread is the concatenation of the ingress node's IP address and a unique identifier field. It should be noted here with utmost importance that every time a node receives a thread and extends it downstream, it may or may not change color of the thread. While extending a thread, the node will change the color of thread, if the next node is a new node and has not been assigned with any color. This thread extends with the changing color. Color of thread will not be changed if the next hop has already been assigned a color in the network for a particular LSP set up (Fig. 5).

2.7. Thread merging

Thread merging is merging of two or more threads to a single outgoing link When LSR 'L' receives a colored thread, and the outgoing thread from LSR 'L' is colored, merging occurs. In this case LSR'L' merges the incoming thread, thus ensuring no message is send downstream. Merging also takes place, if a link has more number of incoming threads.

For a thread to be merged on LSR'L', the following conditions should hold true: (a) LSR'L' should not be an egress node, (b) outgoing Link of LSR'L' should be colored, (c) the hop count for outgoing thread for LSR L should be at least one greater than the hop count of the incoming thread to LSR L, and (d) incoming thread to Link 'L' should be colored.

2.8. Thread stalling, rewinding and withdrawing

When a colored thread is received, if the thread forms a loop, the received thread color and hop count are stored on the receiving link without being extended This is the special case of thread merging applied only for threads forming a loop and referred to as the 'thread stalling', and the incoming link storing the stalled thread is called 'stalled incoming link'. A distinction is made between stalled incoming links and unstalled incoming links.

When a loop-free condition is satisfied and the thread reaches the desired node (destination), an acknowledgement needs to be passed towards the node, where the thread was initially generated. It follows exactly the same path extend the thread in reverse direction and thus it is called rewinding the extended thread. Fig. 3 shows an example of thread rewinding (Fig. 6).

While rewinding, all the parameters are set to be null. In other words, the color of all the threads is made transparent.

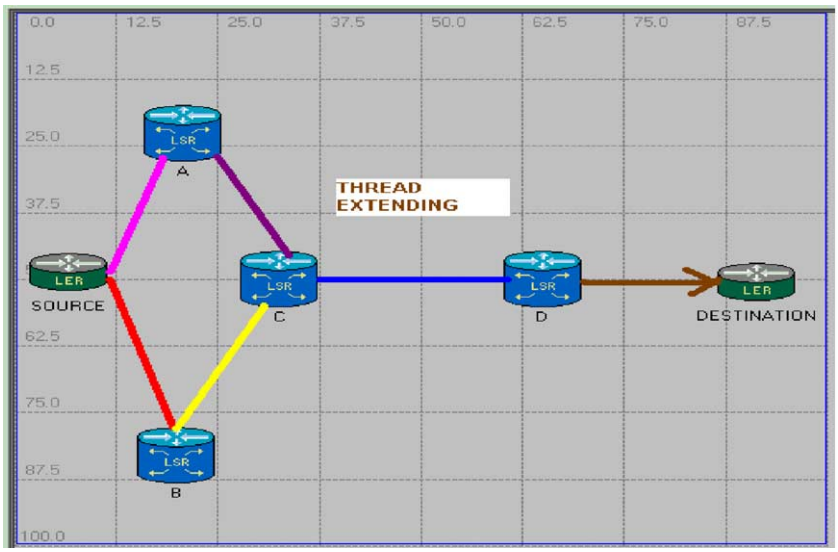

Fig. 5. Thread extending.

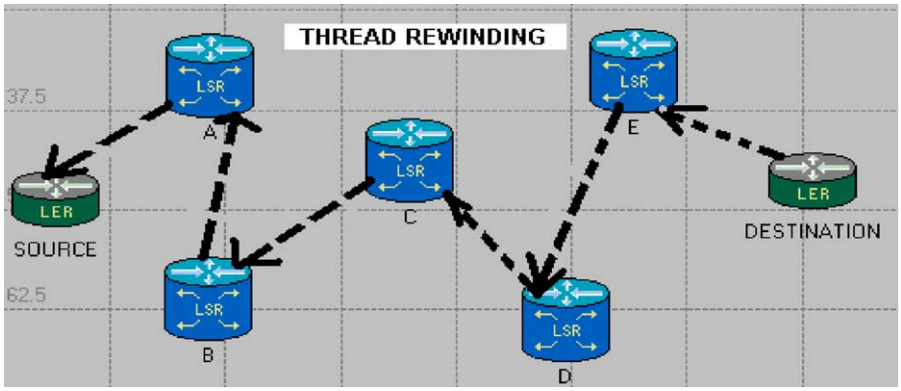

Fig. 6. Thread rewinding.

Furthermore it ensures that the network is ready to be assigned with labels to set up a loop free LSP.

It is possible for a node to tear down a path. A node tears down the portion of the path downstream of itself by sending teardown messages to its next hop. This process is known as the 'thread withdrawing'.

2.9. Loop free condition

The loop-free condition in an MPLS network is: (a) a colored thread is received by the egress node, OR (b) all of the following conditions hold: A colored thread is received by the destination node, Destination node's outgoing link is transparent, AND Destination node's outgoing link hop count is at least one greater than the hop count of the newly received thread.

When a node rewinds a thread, which was received on a particular link, it changes the color of that link to transparent. If there is a link from node M to node N, and M has extended a colored thread to N over that link, and M determines (by receiving a message from N) that N has rewound that thread, and then M sets the color of its outgoing link to transparent. M then continues rewinding the thread, and in addition, rewinds any other incoming thread, which had been merged with the thread being rewound, including stalled threads. Each node can start label switching after the thread colors in all incoming and outgoing links becomes transparent. Note that transparent threads are threads which have already been rewound; hence, there is no such thing as rewinding a transparent thread.

3. Label space

Uniqueness of label is a fuzzy issue A single VPN label may be carried across an entire network, whereas the local label works as a physical next-hop marker. If a standard IP based router can decide internally, where a packet needs to be sent in order to reach a destination, then that router only has to have one address for every router to reach it. Per interface, labels are unique per interface, which means each interface of an LSR has its own label space. Thus, different interfaces of an LSR can use exactly the same label for different bindings. Labels in the interface label space are unique per interface; the same labels can exist in another interface label space while platform labels are unique over the entire router. Platform label space is also referred to as 'global allocation pool'. Most implementations use interface label space because label assignment is a local thing and it does not matter if the same labels exist in different interface label spaces. Platform labels become important in fast reroute link protection where the labels need to be unique on the entire platform (because the label pushed over the back up link needs to be different from the label pushed on the primary link). Labels are always between 1 and 1048575.

4. Applicability of algorithm

The Extended Colored threads Algorithm is applicable for smaller networks such as Intranet subsystems for a huge organization. Instead of IP addresses we used in the colored thread algorithm, we can use global unique variables. RFC 3036 [6] suggests, that when there is no loop detected in a network, the threads are rewound to the point of creation and as they are rewound, the labels are assigned. This proposal goes along with using less memory space within router. By using the unique label, we are just using 2^{20} bits for generating color, instead of 2^{32} bits for IP address. With this all, the known routers within the Intranet have already been assigned a global variable, and since labels are already assigned after loop detection, there is no need to assign it after the algorithm. This clearly means that if the number (n) of routers is in use for setting up the LSP, then we are reducing the total memory usage of the LSR's by the factor of 'n' thus substantially improving the efficiency of Intranet structure. The algorithm has the following logical steps: (1) per Platform Labels are assigned within the network to each router; (2) each router knows, about each and every router within the network; (3) a shortest path or the desired label

path is generated; (4) threads are extended; (5) colors are assigned (color of thread will be unique within a particular cloud); (6) loop free condition is achieved; (7) when thread reaches the destination, it is rewound and While rewinding, Label flag is made 'high' (Label flag 'high' indicates that Label has been assigned, while the thread is rewound); and (8) LSP is set up.

The proposed algorithm performs an efficient way to reduce the memory usage of an individual router by handling fewer bits. Furthermore, the time taken to assign labels for setting up the LSP is saved by assigning labels, while rewinding threads. This work can be extended in several directions. First, this algorithm selects a single shortest path for the router to initiate the thread. Extensions to this algorithm may take into account extensive node failures, multiple links or multiple node failure, or the computation of several backup paths to improve the pliability of the routing path. Second, Implementation of this algorithm on an FPGA chip, can be accomplished and then have a sequence selected for the algorithm to use on requirement basis, e.g. colored thread algorithm or extended color thread algorithm. In addition, it needs to be determined, if the techniques developed for this algorithm for MPLS network gets its place in the Internet besides Intranet with as less memory usage as possible finally, the main extension to this work includes an implementation in commercial routers and deployment in large-scale networks for MPLS routing and traffic engineering.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102–114.
[2] K. Scott, N. Bambos, Routing and channel assignment for low power transmission in PCS, Universal Personal Communications (1996).
[3] V. Jolly, S. Latifi, Towards an efficient loop prevention mechanism, The 2003 International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA (2003).
[4] Development platform for self-organizing wireless sensor networks, Proceedings SPIE, Unattended Ground Sensor Technologies and Applications 3713, 257–268.
[5] Y. Ohba, Y. Katsube, E. Rosen, P. Doolan, MPLS Loop Prevention Mechanism, IETF Networking Group, RFC 3063 (February 2001).
[6] Y. Ohba, Issues on loop prevention in MPLS networks, IEEE Communications Magazine 37 (12) (December 1999) 64–68.
[7] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, IETF Networking Group, RFC 3031 (January 2001).
[8] Paul Brittain Adrian Farrel, MPLS Traffic Engineering: a choice of signaling protocols, Data connection, white paper, 2000.
[9] J. Moy, Applicability Statement for OSPF Protocol overview, IETF Networking Group, RFC 1370 (April 1998).
[10] A. Forcina, A. Luvison, F. Perardi, A strategy for ATM introduction into public networks communications, SUPERCOMM/ICC '90. Conference Record 4 (1990) 1596–1601.