Link State Protocol data mining for shared risk link group detection

G. Das*, D. Papadimitriou**, W. Tavernier*, D. Colle*, T. Dhaene*, M. Pickavet*, P. Demeester*

* Department of Information Technology (INTEC), Ghent University, Belgium, goutam.das@intec.ugent.be, * * Alcatel-Lucent, Antwerp, Belgium

***Abstract*— In this paper, we use machine learning technique at the routers to study the link state protocol data to predict the existence of shared risk link groups (SRLG) in the network. In particular, we use the correlation between different link state updates (LSUs) issued by different network nodes (routers) upon failure. The concerned network router then runs a novel Bayesian network based statistical learning process to learn about the possible existence of SRLGs. The decision of this online learning is transferred to the routing information base (RIB) so that it can accordingly modify the routing table for the entire SRLG upon failure detection of one of the candidate node of that particular SRLG and hence reduce the protection switching time.**

I. INTRODUCTION

The idea of cognitive network has been introduced recently to allow network elements to learn about their own behavior and environment over time, analyze problems, and tune their operation. In such networks, nodes are supposed to learn additional insight from the existing behavior and implement it without any external intervention to improve the network resilience and performance [1-3]. Cognitive networks generally use various machine learning technique to enable the routers to learn online about new network behavior under variable traffic patterns and topological events. The concept behind this idea is to make the network responsive enough to adapt to newer scenario instead of designing it from scratch.

Different kinds of applications that would benefit of machine learning have been studied over the past few years like traffic anomaly detection [4], geography-aware path computation, etc. In this paper, we propose to improve the router recovery time upon topological link failure by identifying the existence of shared risk groups (SRGs). The identification of SRG is performed by applying machine learning technique to link state routing information. SRGs define groups of network resources such as links, and then referred to as Shared Risk Link Groups (SRLG) or nodes, that share a common source of failures and thus likely to fail simultaneously. For example, two distinct logical links routed through the same optical channel or two physical links carried over the same duct would fail altogether in case of optical channel or duct failure, respectively. A set of one or more links fail simultaneously when their associated shared risk occurs. When network layer routing information is exchanged by means of a link state routing protocol, link states are flooded throughout the network: each router advertizes these states to its adjacent nodes by means of link state update (LSUs) message. When a link failure occurs, the routing protocol initiates advertisement of LSUs indicating link state change. SRLG failures often trigger multiple such topological link failures and all the routers in the network take individual decision regarding each of the advertised LSUs. Instead, if the router employs an additional mechanism to learn about the existence of SRLGs from the time sequence of LSUs, then decisions regarding SRLG failure can be taken promptly and simultaneously to avoid additional delay for choosing alternative shortest-paths (across the updated topology). This decision is very handy when network failures are frequent like in wireless sensor networks where multi-hop routing is employed. For ordinary IP network, where the failures are sparse, application of the proposed technique implies log files analysis to learn about SRLGs.

In the literature, the identification of SRLG has recently been considered to be of prime importance in optical networks and especially for IP/Multi-Protocol Label Switching (MPLS) path computation [5-7]. So far, there has not been any method introduced to detect SRLG from the network control traffic such as link state data [5]. The SRLG computation is mostly dependent on manually configured local entries in terms of optical switch or fiber layout [5-6]. These entries are then advertized to the entire network by associating an additional SRLG identifier attribute to the link state advertized by the routing protocol. This SRLG information is then used for SRLG disjoint path computation to improve the network protection and restoration mechanism [6]. The other method to detect SRLG is based on topology and geographical location of physical nodes [7]. However, manually configured entries and geographical information are often difficult to access for legacy equipment which are already deployed in the field. In this paper, we introduce a new machine learning technique to analyze the already available link state protocol data to detect SRLG. Though there has been some attempt to analyze link state protocol data for SRLG identification [8-9], they never considered the statistical information embedded in the link state protocol data to detect distinct SRLGs comprising common members (e.g. a given link part of two distinct SRLGs). In this paper, we introduce a probabilistic algorithm that enables a more complete and accurate detection and identification of SRLG.

The rest of the paper is organized as follows. Section II provides a detailed description of the proposed algorithm. Section III discusses the results and key findings of our algorithm while Section IV concludes the paper.

II. Machine Learning Technique

In this section, we introduce a novel machine learning technique for identifying SRLGs. Before giving further details on the SRLG identification technique, we provide an example of network that will help us to better understand the proposed algorithm.

Fig. 1 shows a typical network topology with 6 nodes (routers) and 9 links. Fig. 2 shows a typical link state update sequence for the same network with links D, B and H failing. For any link state protocol, adjacent routers exchange Hello messages to test the liveliness of their adjacency [10]. Upon timeout, if a particular router does not hear any Hello from its neighbor, it assumes the link connecting two routers has failed. This router then originates a Link State Update (LSU) message and floods it to the entire network to notify every router of the topological link state change (reflecting the link failure).

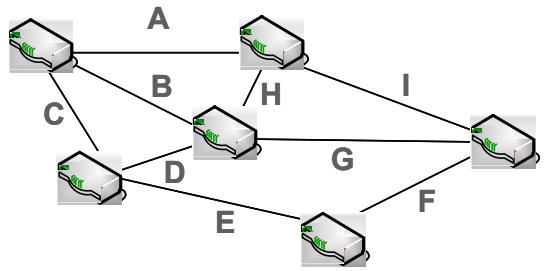

Fig 1: Example network topology

In OSPF for instance, two adjacent routers periodically exchange Hello messages to maintain their adjacency. If a router does not receive a Hello message from its neighbor within a period of time equal to the RouterDeadInterval (that is of the order of seconds), it assumes the link between itself and the neighbor to be down. The detecting router then generates a new Router Link State Advertisement (LSA) to reflect the topological change. Once originated by the detecting router, each LSU packet carries a set of LSAs one hop further away from their point of origination. At each hop, a new LSU is generated that may contain the LSAs of several routers. Any distant router, upon receiving new LSA, installs them in the Link-State Database (LSDB). Using the information of the LSDB as input, a router calculates the shortest-path tree using itself as the root yielding a set of routes stored in the routing information base (RIB). Multiple LSAs for the same link failure can be initiated by different routers. However, observant router can discard those repeated information and consider the first router LSA reflecting the topological change involving a particular link, and thus the LSU that comprises this LSA. With multiple link failures, the arrival time sequence defined by the series of received LSUs is depicted in Fig. 2. To identify SRLGs, it is then required to find correlation pattern among LSU time sequences. We start our algorithm by grouping the links comprised in LSU time sequences and declaring these groups of links as SRLGs. As depicted in Fig.2 for example, the set of links comprised in each of the groups (D, B), (H, D, B), (H, D) and (B) forms SRLGs. Indeed, the temporal locality between the received LSUs is assumed to reflect the simultaneous failure of the links advertized in the LSAs carried in the observed LSUs. We next form a probabilistic model to represent SRLG in a more realistic fashion.

In the next three subsections, we describe the three phases of our proposed SRLG detection and identification algorithm and illustrate it with representative examples.

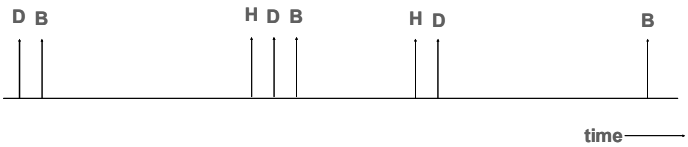

Fig 2: Link state update sequence for different link failure

A. Learning phase

In the learning phase, we construct a new Bayesian network based state space model. An example of state space model is shown in Fig. 3. Each node of the model represents a set of one or more links that forms a particular SRLG. The bottommost tier signifies isolated links forming their own SRLG. As we move higher through the state space, the number of links per SRLG increases. Each of the nodes is connected to one or more upper tier node. These connections specify the transition probabilities; each transition means that an observed SRLG in a particular state space node has a finite probability of having an additional link included in the same SRLG.

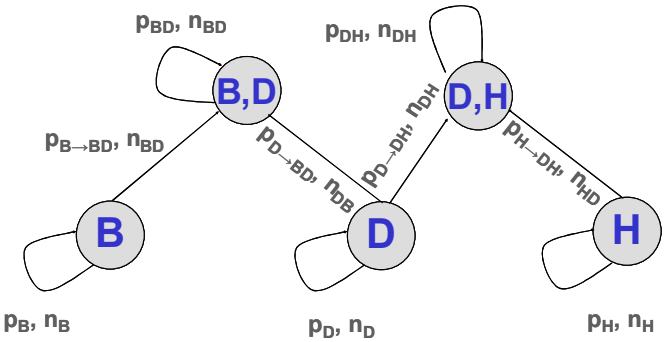

Fig 3: State space model formulation example

In Fig. 3, p_B, p_D, p_H represents the probability of a link (B, D and H respectively) being an isolated link forming their own SRLG. Whereas, state (B, D) and state (D, H) represents two SRLGs with two member links each. In that context, p_{BD} and p_{DH} represents the probability of (B, D) or (D, H) forming SRLG without the possibility of including any further links in their respective SRLGs. The transition probability from state B to state (B, D), $p_{B\to BD}$, represents the probability of finding an SRLG with member links B and D, when an isolated link failure B is initially observed. Each of the transitions in the state space model is associated with a number that indicates the frequency of the observed phenomenon during the learning phase. The first part of the learning phase consists of grouping together LSUs from the observed LSU time sequence. This can be performed in two different ways:

1. We define a time threshold and the LSUs received within the elapsed time interval are grouped together. The time threshold is calculated by summing the maximum synchronization time mismatch between routers, i.e., Hello message synchronization mismatch between routers, maximum propagation delay and queuing delays. These delays may vary from network to network and have to be manually configured as part of the initialization of the machine learning algorithm.
2. We initially start with a time threshold which is referred as window threshold. As its impact on the adaptive algorithm is very limited, the start time can be set to any reasonable estimated time (5 sec. for our case). The window threshold is then adaptively modified as follows. We define T_{min} as the minimum value of timing threshold. The grouping algorithm starts by setting the threshold time (T_{th}) to T_{min}. Then, the algorithm performs one of the following actions if the associated condition is verified:

a) If no more LSA arrives within T_{min}: the algorithm stops further grouping and keeps $T_{th} = T_{min}$.
b) If N number of LSA arrives within T_{min}: the algorithm increases the threshold time as follows: $T_{th} = T_{min} + f(N)\sum_{j=2}^{N} T_{iner\_arrival(j,j+1)}$, where $f(N)$ denotes the function of N and $T_{iner\_arrival(j,j+1)}$ denotes the inter arrival time between j^{th} and $(j+1)^{th}$ arrival.

Under condition (b), i.e., when multiple arrivals of LSAs are observed, the algorithm further waits for subsequent LSA arrivals until the newly computed time T_{th} elapses. In this case, two different scenarios are possible:

b.1) No further LSA arrives when T_{th} elapses: if the function $f(N)\sum_{j=2}^{N} T_{iner\_arrival(j,j+1)} < T_{min}$, then T_{th} is set to T_{min}; otherwise $T_{th} = f(N)\sum_{j=2}^{N} T_{iner\_arrival(j,j+1)}$ and the algorithm stops grouping LSAs. The current threshold time T_{th} is stored for the next phase of grouping.
b.2) N number of LSA arrives within the extended time T_{th}: the threshold T_{th} is then further extended, following the formula defined in point b) above (i.e., N number of LSA arrives within T_{min}), and the algorithm further waits for new LSA arrivals during this extended period.

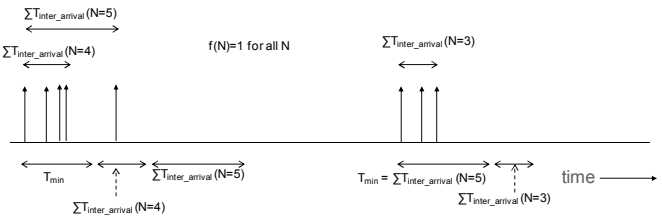

Fig 4: Time sequence of LSA grouping algorithm

The algorithm continues till there are no new LSA arrivals within the extended time T_{th}. The step b.1 in subsequent states provides a mean where the algorithm adaptively reduces the T_{th} value when there are no arrivals within an extended T_{th} period. Whereas step b.2 (and step b) ensures that the algorithm can adaptively increase T_{th} if additional LSA arrivals are observed during the extended T_{th} period. The execution of the algorithm is illustrated by an example in Fig. 4, where we have assumed that $f(N) = 1$ for all values of N.

After the LSU grouping is performed, the state space creation algorithm takes over. During the execution of the state space creation algorithm, each of the states or nodes maintains a number of relations with its upper tier. Each time a relation is registered for a particular state, an identifier (Id) is assigned. We refer to this identifier as the relation Id. Relation Ids are created from the state space name and are shorted appropriately to store so that the overall search time to find the existence of a relation is minimized. Each transition from a given state to its upper tier state is associated with four variables namely, (i) the transition identification, (ii) the frequency of the observed phenomenon during the learning phase, (iii) the transition pass, that indicates the number of similar probabilistic transition a node can have to its higher tier, and (iv) the associated probability for that particular transition. Every time a transition is observed, the algorithm searches among the existing relation Ids assigned to already registered relations for that particular state. If no match is found, a new relation is created; otherwise, the frequency counter for that relation is increased by a value set to one. For a particular group, the pass number indicates the depth of the state from its starting state. Suppose for instance that the group (B, D, H) is detected from the LSU grouping. Initially, a state of (B, D, H) is created. Then, at the next lower tier, three groups of (B, D), (D, H), (B, H) are created. Here, the pass number automatically becomes equal to 1 for each of the transition from these three groups to the higher state (B, D, H). Moreover, we can form a next tier of (B), (D) and (H) state, whose pass becomes 2 as they are two layers deeper into the SRLG tree for current observation. The important point is that for a particular relationship, a separate counter is maintained for each pass value. This particular counter retains how many times this relation has been observed. We can also observe that the value of the transition pass corresponds to the total number of transition from the concerned state to all possible next tier state.

The transition probability is defined by the following formula, where $pass(k)$ is the pass number associated with k^{th} pass for a particular relation and $\eta_{pass(k)}$ is the number of times a given pass has been observed by the learning phase. The value of k can take any values from 1 to n-1.

$$p_{transition} = \frac{\eta_{pass(k)}/pass(k)}{\sum_{all\ pass}(\eta_{pass(k)}/pass(k))} \quad (1)$$

We provide the following example to illustrate the above algorithm. Here, we present the evolution of the state space model resulting from the arrival of the first two groups as depicted in Fig. 2.

1. State space formation due to the detection and the identification of the first group (B, D) from the arrival of the LSU sequence, as shown in Fig. 5.
2. State space modification due to the due to the detection and the identification of the second group (H, D, B) from the arrival of the new LSU sequence, as shown in Fig. 6.

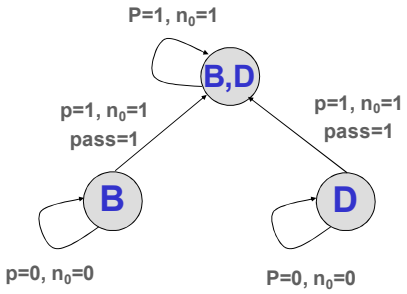

Fig 5: State space model for the first group (D, B) of Fig. 2

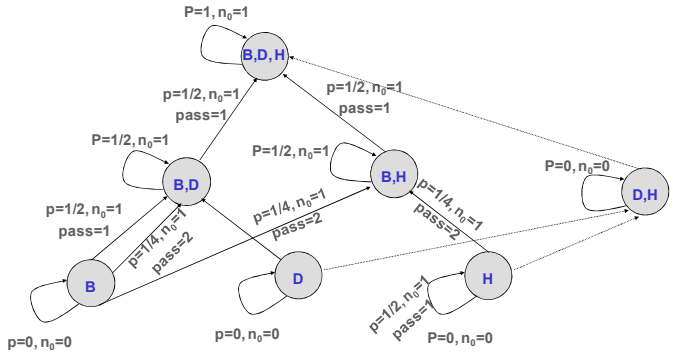

Fig 6: State space model for the second group (H, D, B) of Fig. 2

Initially, as depicted in Fig. 5, the transition pass number is set to 1 for both the transition and the self-transition probability for state (B, D) is 1. However, the self-transition probability for states (B) and (D) remains equal to zero. For the second group (H, D, B) the value of the self-transition probability changes to 1. The pass number for transitions from the second tier ((B, D), etc.) to the upper tier (B, D, H) is set to 1. However, a new transition with a new pass number 2 has to be created between first tier states ((B), (D), etc.) and the second tier states (B, D), etc.). Therefore, the associated probability values for each of the transitions are modified according to Equation (1). Note that transitions are directed in the upward direction only. The total sum of the transition probabilities going out of a node always sums up to 1, which verifies the validity and correctness of the algorithm.

Now, in order to provide a general description of the above described algorithm, we suppose that there are n links denoted by l_1, l_2, …, l_n where l_1, l_2, l_3, … can respectively represent link A, B, C, … of our example scenario.. We denote the SRLG set as $S_{p,q}$, where p specifies the number of member link within that SRLG, and q specifies the SRLG id for individual SRLGs having p specified links chosen from the pool of n links as group member. State $S_{p,q}$ for different values of q can represent any state of p^{th} tier in our state diagram like (B, D) or (B, H) for our example state space model.

As the total number of links is equal to n, the highest number of links an SRLG can comprise is denoted by $S_{n,1}$, where index 1 specifies the uniqueness of that particular SRLG. Note that only one SRLG is possible with all the n links as member. The next hierarchy of SRLG (in terms of number of links as member) can be represented as $S_{n-1,q}$ where q can vary from 1 to ${}^{n}C_{n-1}$, i.e., the all possible number of combinations with which $n-1$ links can be chosen from n links. The further hierarchy and their properties are specified in the table below:

Hierarchy number	Symbol	Possible value of q	Member links per SRLG	Subset relationship
0	$S_{n,1}$	1	n	
1	$S_{n-1,q}$	1, 2, …, ${}^{n}C_{n-1}$	n-1	$S_{n-1,q} \epsilon S_{n,1}$
		…		
R	$S_{n-R,x''}$	1, 2, …, ${}^{n}C_{n-R}$	$n-R$	$S_{n-R,x''} \epsilon S_{n-R+1,\tilde{x}''}$ *Where* $\tilde{x}''$ *can take any of R values for which SRLG* $S_{n-R+1,\tilde{x}''}$ *contains all the members of the group* $S_{n-R,x''}$
		…		
r	$S_{n-r,x'}$	1, 2, …, ${}^{n}C_{n-r}$	$n-r$	Similar as hierarchy R
r+1	$S_{n-r-1,x}$	1, 2, …, ${}^{n}C_{n-r-1}$	$n-r$-1	Similar as hierarchy R
		…		
n-1	$S_{1,q}$	1, 2, …, ${}^{n}C_{n-1}$ or n	1	Similar as hierarchy R

Table 1: SRLG state hierarchy and their specification

Table 1 describes states with $n-R$, $n-r$, $n-r-1$ and 1 links as member link of a particular SRLG group. The identifiers of such states are represented by x', x'', x and q respectively. We also specify the possible relationship of these states with the immediate next upper tier states in the hierarchy. For example, a states in the hierarchy number R (i.e, $S_{n-R,x''}$) can have R number of relationship with $S_{n-R+1,\tilde{x}''}$ as there can be R groups in the $R\text{-}1^{th}$ hierarchy for which all the member links of $S_{n-R,x''}$ are members. Referring to our algorithm, for each state of the form $S_{n-r-1,q}$, a state variable and a structure of relationships with the members of r^{th} hierarchy number are maintained. The structure provides the relationship of $S_{n-r-1,q}$ to the immediate upper hierarchy i.e., $S_{n-r,x'}$ and to itself. For each of the relationship, there can be multiple passes depending upon the starting point of the SRLG group (e.g., $S_{n-R,x''}$) as shown in the examples depicted in Fig.5 and Fig.6 (e.g., group (B, D, H) or group (B,D)). This explains why the hierarchy number R is introduced as part of Table 1. The passes are denoted by *pass(k)*, where the maximum value of the number k specifies how many tiers below the concerned state (e.g., $S_{n-r-1,x}$) exist from the starting state (e.g., $S_{n-R,x''}$) in the state space model. Therefore, max. of $k = (n\text{-}r\text{-}1)\text{-}(n\text{-}R) = R\text{-}r\text{-}1$ for a relationship between $S_{n-r-1,x}$ and $S_{n-r,x'}$ where the starting SRLG for this learning phase is denoted by $S_{n-R,x''}$. Each of the structure under the same pass keeps a counter ($\eta_{pass(k)}$) that is incremented after every occurrence of such relationship. Every relationship is also associated with a transition probability value that is updated upon completion of

each iteration of the learning algorithm according to Equation (1).

This concludes the learning phase of our algorithm. In the learning phase, we create a state space representation of the SRLG map where the transition probabilities provide a statistical estimator of whether a given group should be considered as an SRLG or not. Learning can be performed offline, fed with the previous log file stored regarding network failures. The machine learning model can be programmed to filter out data from the log file to create LSU update sequence and hence, initial LSU groups which the algorithm can take as input data. In addition, the learning phase may continue functioning online while link failures occur. Additional observations can then dynamically modify the structure of the SRLG state space and the associated transition probability values. Irrespective of the mode the learning phase is executed, the algorithm for SRLG detection needs the next phase of deciding and declaring SRLGs from the probability values provided by the learning phase.

B. Decision making phase

In the decision making phase, we use the state space based model and its transition probabilities to construct a control scheme regarding the existence of SRLGs in the entire network. The decision making process follows the following recursion.

Whenever a new LSU arrives mentioning a link failure, it triggers the following probability computation. The algorithm detects all the transitions between the state representing the failed link and all other states part of the state space model. We need to keep in mind here that the transitions are only possible from a lower tier state to a higher tier state. The algorithm eventually computes the overall probability of reaching all the possible upper tier states that is connected to the concerned state with a defined transition probability. Probability values are then compared to a pre-defined threshold. The highest tier state that has a probability higher than the pre-defined threshold value is then declared as the representative of SRLG. With the state space model depicted in Fig. 6 (using Equation 1):

p(BD | B) = p(B→BD)(for pass 1) + p(B→BD)(for pass 2)
= 1/2+1/4 =0.75
p(BH | B) = 1/4 = 0.25
p(BDH | B) = p(B→BD)p(BD→BDH)+p(B→BH)p(BH→BDH)
= 0.75*0.5+1/4*1/2 = 0.5

Using this example, we can observe that if the threshold probability is lesser than 0.75 and greater than 0.5 the SRLG include link B and D, whereas if the threshold probability is below 0.5, it chooses B, D and H as the link members of the SRLG. If two (or more) states part of the same tier have an equal probability and are the highest states having probability above threshold, the algorithm defers decision and waits for additional inputs.

In a more general term, the transition probability computation can be summarized by means of Equation (3). Here $pr(S_{n-R,x''}/l_\alpha)$ denotes the overall probability of having an SRLG represented by $S_{n\text{-}R,x''}$ given a link failure related to l_α arrives where l_α belongs to $S_{n\text{-}R,x''}$ set.

$$pr(S_{n-R,x''}/l_\alpha) = \sum_{\substack{\left(\prod_{m=1}^{n-R-2}\binom{m+1}{m}\right) \text{ combination of} \\ \text{transition for differnt } i,i' \\ \text{where } S_{j,i}\subset S_{n-R,x''} \text{ and } S_{j+1,i'}\subset S_{n-R,x''} \\ \text{for all } j<n-R}} \left(\prod_{j=1}^{n-R-1} pr(S_{j,i}\text{-}\triangleright S_{j+1,i'})\right) \quad (2)$$

Here $pr(S_{j,i}\rightarrow S_{j+1,i'})$ is calculated as follows:

$$pr(S_{j,i}\text{-}\triangleright S_{j+1,i'}) = \sum_k pr\left((S_{j,i}\text{-}\triangleright S_{j+1,i'})|pass(k)\right) \quad (3)$$

Equation (3) states that the probability associated to a particular SRLG represented by $S_{n-R,x''}$ from a particular link l_α, represented by the state $S_{1,\alpha}$, depends on all possible transitions through the state space model from $S_{1,\alpha}$ to $S_{n-R,x''}$. The summation term in Equation (3) considers all possible ways to connect state $S_{1,\alpha}$ to $S_{n-R,x''}$. It is a trivial exercise to show that there exist $\left(\prod_{m=1}^{n-R-2}\binom{m+1}{m}\right)$ ways to connect (track down from) state $S_{n-R,x''}$ to state $S_{1,\alpha}$.

It is evident that the selection of the decision probability threshold is crucial for the overall performance of the proposed algorithm. We propose to run the same algorithm offline on the log file of LSU sequence of some router in a network with known SRLG groups for varying values of probability threshold to find out the optimized value of the probability threshold.

This concludes the description of the decision making phase. In the next subsection, we describe the use of the proposed SRLG identification mechanism to reduce recovery time.

C. Protection time reduction phase

In this phase, we use the outcome of the SRLG identification phase to reduce recovery time upon failure occurrence. As discussed earlier, identification of SRLG may trigger simultaneous routing and forwarding table updates for multiple links failure scenario with the arrival of single link failure notification. Modification of routing and forwarding table might take several milliseconds depending on the router entry. With any of the current link state protocol (such as OSPF), common link failure resulting from an "SRLG failure" may trigger multiple LSDB update and subsequently RIB entries re-computation, one to address each of the link failure. Failing to prune the set of links involved by the SRLG failure at RIB entries re-computation leads to higher magnitude of packet losses compared to the situation where the set of links (associated to this SRLG failure) result in a single re-computation step. It has to be noticed that, independently of the actual RIB entries re-computation time, failing to take into account the set of links affected by the SRLG failure leads to traffic losses until all failed links have been pruned from the LSDB used as input for RIB entries re-computation.

We here briefly describe the architecture for enhancing a legacy router with a machine learning (ML) component. As

shown in Fig. 7, the ML component or engine interacts with the existing routers engines. Such router comprises four basic modules: the machine learning engine (MLE) and the monitoring engine (ME) in addition to the ordinary forwarding engine (FE) and routing engine (RE). The monitoring engine is not involved in the present application and thus not further considered in this paper. The SRLG identification algorithm runs in the MLE. The typical data flow diagram is shown in Fig. 7 with numbered arrows.

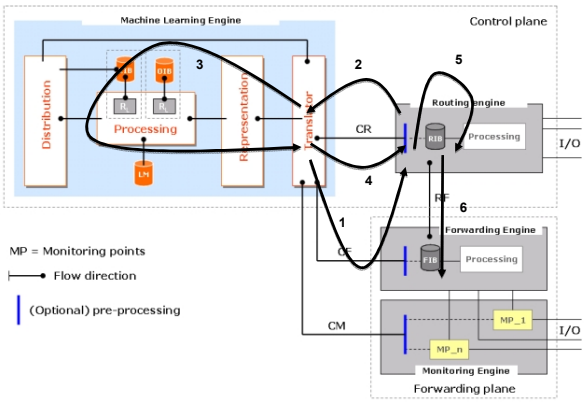

Fig 7: Router architecture along with SRLG decision flow diagram

The functionality of each of this data flows are described as follows:

1. MLE initiate request for link state updates from the RE.
2. RE starts sending update to MLE as they are available. RE continues to do so as the algorithm runs.
3. MLE learns and detects SRLG with Bayesian Network based state space model.
4. MLE sends SRLG prediction to RE.
5. RE re-calculates routing table for SRLG failure.
6. RE updates forwarding information base (FIB) entries for SRLG failure.

Therefore, the described architecture with the SRLG identification algorithm present in the MLE helps router to reduce protection switching time.

We now provide a brief description of the computational complexity involved in our proposed algorithm to prove that our algorithm is feasible from computational point of view. To calculate the total number of states, we find ${}^{n}C_{n-t}$ states for each tier (t) of our hierarchy as described in Table 1. Therefore, the total number of states in the worst possible case (where all the links in the network form an SRLG) is given by the following formula: $\sum_{t=0}^{n} {}^{n}C_{n-t} = (1+1)^n = 2^n$. For a very large network, with a large number of links, this might lead to an explosion of the number of states. In order to compute the number of relationship to be maintained for each state, we observe that each of the states in tier t maintains t relationship with its immediate upper tier, i.e, tier t+1 (see Table 1). Therefore, the total number of relationship to be maintained in the worst possible case is $\sum_{t=0}^{n} {}^{n}C_{n-t}t = n2^n$. Hence, the memory requirement will be proportional to $n2^{n-1}$, which might be difficult to realize for a very large network. Similarly, for the worst possible case, the learning and decision making phases might require computation complexity of the order $O(n2^{n-1})$. However, if we assume that there are s SRLGs present in the network with $u_1, u_2, \ldots, u_s$ links per SRLG respectively, memory requirement with our algorithm is reduced to a number proportional to $\sum_{j=1}^{s} u_j 2^{u_j-1}$, which is much lesser than $n2^{n-1}$ (considering $n = u_1+u_2+..+u_s$). This memory reduction can be achieved when the initial grouping algorithm described in the first part of Section II.A provides the best possible results. However, we argue that the numbers will remain very close to the theoretical bound as failures are rare events and therefore the probability of wrong grouping is sufficiently low. The computational complexity for a link within an SRLG group with link number u_j is of the order of $o(u_j 2^{u_j-1})$. If a link is member of g SRLG, the computational complexity increases to $o(\sum_g u_j 2^{u_j-1})$, which still remains reasonably low.

III. Experimental Results and Discussion

Our experimental setup is based on an example network with predefined topology and SRLGs. We create an approximate analytical framework to represent the time sequence of LSUs for such a network. In this section, we first describe how we formulate and generate the input LSU time sequence to our state space based model. We use **Matlab** [13] to generate these time sequences and then we run our algorithm on these time sequences to validate our model.

As obtaining failure data set is difficult (due to the fact that failures are relatively sparse events), we created a model to generate input data set that closely mimics the real scenario. For this purpose, we assume that the link failure or any SRLG failure follows Weibull distribution [11]. The Weibull distribution is known for long for its special property to represent failure scenario [11]. The PDF of Weibull distribution is shown in equation (4):

$$f(x;\lambda,k) = \begin{cases} (k/\lambda)\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^{k}} & for\ x \geq 0 \\ 0 & otherwise \end{cases} \quad (4)$$

Where, λ is the scale parameter and k is the shape parameter. When we take k>1, the failure rate increases with time, which resemble the realistic network scenario. For our purpose, we take k =1.5 and λ a very higher value to counter the fact that network failure events are rare in reality.

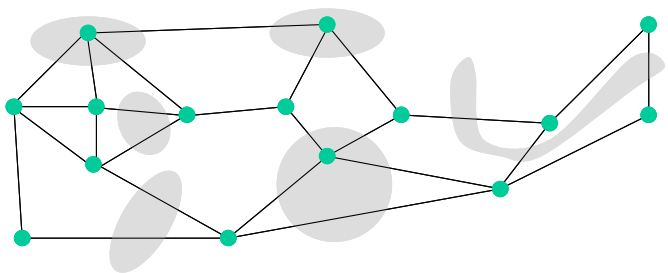

Fig 8: Example network with 25 links and associated SRLGs

We have taken a network of 25 links as shown in Fig. 8

among which links form multiple SRLGs of different link numbers. For our purpose, we have taken one SRLG each with 5 and 4 member links, two SRLGs with 3 and 2 member links and rest forming no SRLGs (as depicted in Fig. 8 with shaded region). We assume all SRLGs are due to the sharing of the same duct. We distinguish two cases depending on the dominant delay factor in the origination and propagation of LSUs after the occurrence of correlated failure:

Case 1: Hello message and RouterDeadInterval detection dominates. Suppose that two distinct LSUs for two different links associated to the same SRLG, are originated by two different routers. The maximum delay these LSUs can suffer at the origin is due to the desynchronization of Hello messages. Indeed, these periodic messages are originated independently by each router within a time interval that ranges in the order of seconds (typical values 1 to 10 sec.). At the higher inter-arrival delay between two LSUs resulting from the same SRLG failure, the Hello interval desynchronization dominates the propagation or queuing delay of LSUs. So for higher inter-arrival delay, we can assume a uniform distribution for the LSU inter-arrival time because the origination of Hello messages by routers is completely random and independent between routers.

Case 2: Queuing, Transmission, and/or Propagation dominates. When the desynchronization due to Hello message is negligible, LSUs propagation and queuing delays dictate the cause for delay variance. Due to the accumulation of multiple mutually independent random queuing delays, we can assume without the loss of generality, that the inter-arrival time between LSUs are memory less and demonstrate exponential distribution. We have tested different fat tailed distribution that have power law decay to verify if any significant changes occur in the final results. However, different inter-arrival time distribution shows minimal effect on the final result because failures are rare events that are usually separated by a large amount of time. Generally, inter-failure occurrence time is much larger than the inter-arrival time between two LSUs for same SRLG failure to create any impact whatsoever.

With this specifically generated input data file that include the time sequences of LSU arrivals, we run the LSU grouping algorithm as described in Section II.A. We use both grouping techniques described in Section II.A. However, very little difference in terms of final outcome was observed. Once the LSUs are grouped, we run the proposed state space based learning model as well as the decision making algorithm to predict the existence of SRLG. We compare our results from the prior knowledge of SRLGs from our assumed network map and compute the amount of false positive and false negative the algorithm generates.

Fig. 9 provides the results for percentage of false positive and false negative as the number of failures per SRLG increases. This percentage value is the overall average among multiple random experiments performed after a certain number of failure data from each SRLGs are used to learn the state space. This experiment was carried over a set of disjoint SRLGs where the members of each SRLG are not part of any other SRLGs.

We next follow the same experiment for SRLGs with common members. Fig. 10 provides the results for multiple SRLGs having one common member. For this example we have included a common member between SRLG group having 5 and 4 members. Therefore, the previous SRLG with 4 members has now 5 member links. We can easily see that the performance degrades due to the interconnection between SRLGs. It can be observed that the algorithm produces both false positive or false negative more often when the SRLG failures with common member link are happening. However, the percentage of false positive or negative decreases as the number of failure per SRLG increases. This decrease in percentage can be explained as follows: as the algorithm gathers more statistical input regarding different SRLG failures, it progressively learns to better predict future occurrences. We further increase the number of common members. We add one more common link to the SRLG with 5 members so that the new SRLG with 6 and 5 links have two common member links. From Fig. 11, we observe further degradation due to two common members between SRLGs.

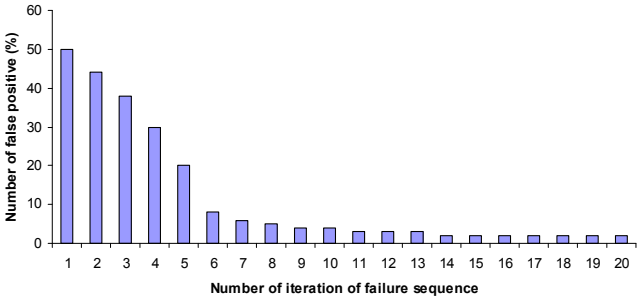

Fig 9: Percentage of false positive and negative with number of failure iteration (disjoint SRLGs)

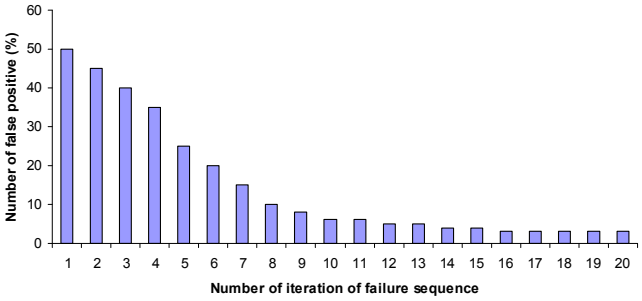

Fig 10: Percentage of false positive and false negative with number of failure iteration (SRLGs with one common node)

Finally, we investigate the gain of our SRLG detection method in terms of reducing the protection switching time and eventually reducing the amount of packet loss during failure. We assume the router update process to follow the quantum update procedure as described in [12] with update time = 100 µs, distribution time = 100 µs, swapping time = 1000 µs, prefixes per batch = 500 and total number of flows through the concerned router = 5000 having flow rate varying from 1 Kbps to 100 Mbps. For our example network with 20 failure cycles the total amount of data loss in the concerned router becomes 16 GBytes without the SRLG detection procedure. Whereas, SRLG detection and reduce it to 7.6 GBytes

(>50%). This is significant for networks where failures are more frequent like wireless sensor networks or networks with old equipments.

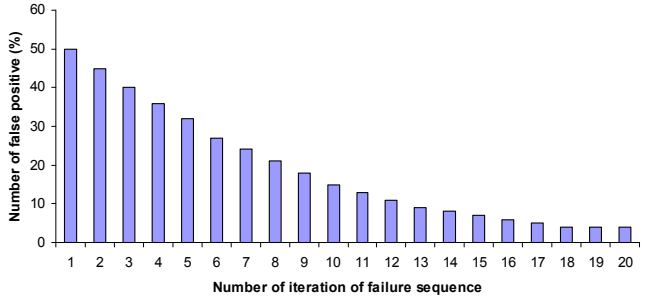

Fig 11: Percentage of false positive and false negative with number of failure iteration (SRLGs with two common nodes)

We intend to further investigate the topology dependency of the proposed algorithm, and in particular, the effects of the network diameter, the network node degree distribution and the clustering coefficient. We see these topology metrics as the opportunity for the future development and evaluation of the proposed algorithm.

IV. Conclusion

In this paper, we have proposed a new Bayesian network based statistical learning process to learn about the possible existence of SRLGs in the network. Our proposed state space model identify SRLGs from the LSU data sequence and help routers to simultaneously modify routing as well as forwarding table for the entire link group forming an SRLG in one go. Thus, our proposed scheme improves upon the protection switching time.

The simulation results show that the SRLG prediction resulting from the execution of our algorithm provides enough confidence. Nevertheless, further testing the role of temporal correlation between different LSUs to detect SRLGs would be beneficial to improve predictive value of the proposed model. We believe the consideration of correlation between LSUs will improve the SRLG detection probability and will reduce the false positive and false negative percentage of our algorithm. We are also trying to enhance our algorithm to better optimize the time threshold and probability threshold values to achieve better accuracy for SRLG identification. For this purpose, we plan to emulate the SRLG failure detection environment and test our algorithm in near real time scenario.

Acknowledgement

This research work is (partially) funded by the European Commission (EC) through the ECODE project (INFSO-ICT-223936) part of the European Seventh Framework Programme (FP7).

References

[1] R.W. Thomas, L.A. DaSilva, A.B. MacKenzie, “Cognitive networks”, in Proceedings of the First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, Baltimore, MD, USA, November 8–11, 2005.

[2] C. Fortuna, M. Mohorcic, “Trends in the development of communication networks: Cognitive networks”, Computer Networks, 2009.

[3] Q. Mahmoud, “Cognitive Networks: Towards Self-Aware Networks”, John Wiley and Sons, 2007, ISBN 9780470061961.

[4] V. J. Hodge, J. A. Austin, “A Survey of Outlier Detection Methodologies,” Artificial Intelligence Review, 22, Kluwer Academic Publishers, 2004, pp. 85–126,.

[5] CY Lee, A. Farrel, S. De Cnodder, “Exclude Routes - Extension to RSVP-TE,” IETF RFC 4874, April 2007.

[6] P. Sebos, J. Yates, D. Rubenstein, A. Greenberg, “Effectiveness of Shared Risk Link Group Auto-Discovery in Optical Networks,” Optical Fiber Communication Conference and Exhibit, 2002.

[7] G. Xiao and X. Pan, “Heuristic for the maximum disjoint paths problem in wavelength-routed networks with shared-risk link groups” Journal of Optical Networking, Vol. 3, Issue 1, pp. 38-49.

[8] R.R. Kompella, J. Yates, A. Greenberg, A.C. Snoeren, “Detection and localization of network black holes,” in Proceedings of IEEE INFOCOM, 2007.

[9] R.R. Kompella, J. Yates, A. Greenberg, A.C. Snoeren, “IP Fault Localization Via Risk Modeling,” NSDI’05: 2nd Symposium on Networked Systems Design & Implementation.

[10] R. Perlman, Interconnections: Bridges, Routers, Switches, and Internetworking Protocols, Second Edition, Addison Wesley Professional Computing Series, October 1999.

[11] Papoulis, Pillai, “Probability, Random Variables, and Stochastic Processes,” Fourth Edition, ,McGraw-Hill, December 2001.

[12] W. Travenier, D. Papadimitriou, D. Colle, M. Pickavet, P. Demester, “Optimizing the IP router update process with traffic-driven updates,” Design of Reliable Communication Networks (DRCN) 2009, Washington D.C., USA, October 2009.

[13] “MathWorks – MATLAB and Simulink for Technical Computing” – www.mathworks.com.