A Novel Scheme for Node Failure Recovery in Virtualized Networks

Habib Abid habid091@uottawa.ca
Nancy Samaan nsamaan@eecs.uottawa.ca
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa - CANADA

***Abstract*—This paper addresses the problem of recovering virtual networks (VNs) affected by a substrate node failure. A novel heuristics-based algorithm that efficiently reallocates new resources for the affected VNs after a node failure is proposed. In this algorithm, a manager substrate node executes a set of recovery steps to migrate all the hosted virtual nodes in the failed substrate node in addition to the virtual paths traveling across it. The proposed approach is executed in a distributed manner without any coordination from the central Infrastructure Provider (InP). The developed scheme efficiently minimizes the node failure recovery cost, the time needed to recover the virtual nodes hosted on the failed substrate node and hence significantly reduces the service interruption period. This, in turn, results in increasing the service provider revenue and decreasing the penalty charges paid for service level agreement (SLA) violation. Performance results demonstrate the significant reduction in VN service cost and interruption time.**

I. INTRODUCTION

The Internet's stunning success has changed the way that people communicate and provided the effective environment for a multitude of business. However, the necessary Internet growth has been, unfortunatly, bounded by reliability constraints and social-economic factors [2], [6], [10], [15]. The implementation of any new architecture needs consensus between ISPs. Network virtualization is introduced as a promising attribute to overcome the above challenges and facilitate service deployment of the future inter-networking environment [10], [11], [15]. Network virtualization is a powerful paradigm that allows multiple heterogeneous virtual networks (VN) to efficiently share the resources in a single physical network infrastructure.

A major challenge in network virtualization context is how to efficiently use the substrate resources (i.e., nodes CPU and links bandwidth). Mapping of virtual nodes and virtual links onto the substrate resources is known to be an NP-hard problem [4].

Several research efforts [5] [3] [19] addressing this challenge have been presented; a number of these efforts introduced different heuristics to solve the problem of VN mapping in the hopes of establishing efficient use of the substrate resources. In addition to the heuristics needed to efficiently map the VNs in the substrate network, we need techniques that manage the resources already allocated to active VNs. Unfortunately, the literature still lacks such techniques. For instance, the infrastructure provider (InP) may need to perform maintenance tasks for some substrate nodes and this will require all hosted VN nodes to be migrated to other nodes in the physical network. Clearly, if the service interruption time due to migration is too long, this operation will cause service level agreement (SLA) violation. Furthermore, solving the problem of efficient mapping of VNs in the substrate network (SN) without taking into consideration the effect of substrate node failure could decrease the InP revenue. Hence, there is a need to develop a technique that can relocate the already hosted virtual nodes in case of node failure or node maintenance while minimizing the service disruption period.

In fact, any VN mapped in the physical network has to be reliable in order to offer its service to the clients without any (or minimized) service interruption. Butt et al. [3] present algorithms for re-optimizing and re-embedding initially rejected VN requests after fixing their bottleneck resources.

While path adaption for VNs has been addressed by a number of approaches [5][13][17], to the best of the authors knowledge, the problem of node failure in VNs has not been addressed before.

In this paper, we introduce a novel distributed node recovery algorithm that efficiently reembeds the virtual nodes affected by a failed substrate node. The proposed scheme relies on the cooperation of set of distributed managers hosted on a number of substrate nodes to achieve this task. The process is triggered by the arrival of a substrate node failure message from the InP. A designated manager node sends a request to the substrate nodes hosting the neighbour nodes of the affected virtual nodes to search for a new candidate substrate node. The search is performed by constructing a shortest path tree (SPT) from the neighboring nodes to all candidate nodes within a specific distance in the SN. The calculated SPTs are then employed to find the optimal candidate node. The proposed work efficiently reduces the node failure recovery cost, experienced delay and service interruption time during this process while maximizing the InP revenue.

The rest of this paper is organized as follows. In Section II related work is presented to review the literature and show how our work is different from existing research. Section III describes the adopted network model and formulates the node failure problem. In Section IV, the proposed substrate node failure recovery algorithm is presented. Section V describes node failure recovery for overloaded networks. Performance

978-3-901882-50-0

results are presented in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

There is a large number of research efforts in the literature that presented approaches to reoptimize resource usage of a substrate network (SN) hosting VNs [3] [13] [17]. The main focus of the majority of these efforts was to relocate a complete VN that uses a node or link with high stress [19], or to perform a reconfiguration for part of the VN (star moving candidate) to reduce the chances of VN requests rejections caused by overloaded substrate links [5].

For example, in order to improve the utilization of the substrate network, two approaches [13][17] presented two techniques of virtual links relocation. Raihan et al. [13] introduced a survivable virtual network embedding (VNE) heuristic that includes a substrate link failure recovery in VNE and proposed a hybrid policy heuristic. Their heuristic proactively computes a set of backup detours for each substrate link before the mapping of virtual network (VN) and uses them later in case of any substrate link failure. Yu et al. [17] developed new notions of optimization of substrate links by allowing the substrate link to split a virtual link between different substrate paths and periodically migrate paths over the substrate network to better utilization of substrate resources. However, the authors didn't take the failure of substrate nodes into account when developing their algorithm.

Fajjari et al. [5] presented a novel virtual network reconfiguration algorithm (VNR). The purpose of their approach is to reduce the number of overloaded (bottleneck) substrate links in order to decrease the rejection rate of future virtual networks. When the virtual network is rejected, the algorithm relocates the star moving candidates with the highest overloaded links in the substrate network in order to minimize the number of congested substrate links and increase the chance of accepting new virtual network requests. The authors developed their approach based on migrating some virtual nodes with their direct connected links to free up some bandwidth capacity in the overloaded links.

Butt et al. [3] introduced techniques for differentiating different resources (links and nodes) based on their criticality on the connectivity of the substrate network and their utilization plus the number of the VNs using them. Furthermore, they proposed algorithms for reoptimizing the already embedded VN requests in order to avoid the fragmentation of the network and improve the overall acceptance ratio, which in turn increase the revenue. Their optimization algorithms include the detection of bottleneck links and nodes and relocation of nodes and links to release some capacity in bottlenecks.

In [19], Zhu et al. presented the reconfiguration algorithm VNA-Periodic. This algorithm reconfigures only a subset of the existing VNs that are most critical in reducing the overloaded nodes and links instead of reconfiguring all existing VNs. This approach operates mainly in two steps. In the first step, it marks a set of virtual networks hosted on the substrate network that use at least one overloaded physical link (or node). In the second step, VNA-Periodic uses the VNA algorithm adopted in the same paper to relocate the marked virtual networks.

Although all studies above attempt to improve the VN mapping process, none of them solves the problem of resource reallocation after a node failure.

However, there are some efforts in the literature that provide self-repair algorithms to have the node failure recovered in overlay networks. Porter et al. [12] presented an algorithm that run a decentralized repair of failed nodes in overlay networks, and this algorithm can be seen as a framework used by different mechanisms. The authors designed a three-phases repair technique: (1) determining the extent of the crashed section (2) identifying the repair coordinator (3) executing the repair procedure. Also, Stoica et al. [14] developed Chord, a distributed lookup protocol, that deals with nodes joining the system and with nodes that fail or leave dynamically. The authors presented a technique by which redundantly stores data on another nodes, and ensures that requests for data on failed nodes are redirected to nodes having backups. Furthermore, Jannotti et al. [8] developed a protocol for constructing efficient and scalable distribution trees that adapt to changes in the conditions of the substrate network. It introduced a system that maintains a tree structure for the sake of the node failure. The system uses a technique that enable child nodes maintain ancestor lists, which in turn used to locate and attach to a surviving ancestor if their current parent fails.

In this paper, we focus on SN node failure and introduce a scheme that can minimize the cost and time needed to re-embed the virtual nodes affected by node failure in a decentralized manner.

III. NETWORK MODEL AND PROBLEM DESCRIPTION

A. Network model

We model the substrate network (SN) as an undirected weighted graph denoted by $G_s = (N_s, L_s, R_s)$, where N_s is a set of substrate nodes, and L_s is a set of substrate links. Each substrate node $n_s \in N_s$ is characterized by its CPU capacity which is denoted by $cpu(n_s)$, and each substrate link $l_s \in L_s$ is associated with its bandwidth capacity $bw(l_s)$. R_s represents the set of manager nodes, where $R_s \subseteq N_s$, that are responsible for running the proposed node failure recovery mechanism. Each manager node is associated with one or more SN nodes in the physical network and performs the steps necessary to recover from node failure. This assignment is decided by the InP. The process of assigning the manager nodes and the criteria considered in the selection of the manger nodes will be investigated in future work.

A virtual network request (*VN*) can be represented by an undirected weighted graph $G_v = (N_v, L_v)$, where N_v is the set of virtual nodes n_v associated with their CPU requirements $cpu(n_v)$ that must be satisfied, and L_v is a set of virtual links l_v, each of which has its bandwidth demand $bw(l_v)$ to be met in any VN mapping.

TABLE I: The maintained table by manager node 9 in Fig 1

VN ID	Virtual node	Capacity	Adjacent nodes
1	a	20	x hosted on node 4 y hosted on node 8
2	b	20	f hosted on node 4 c hosted on node 1

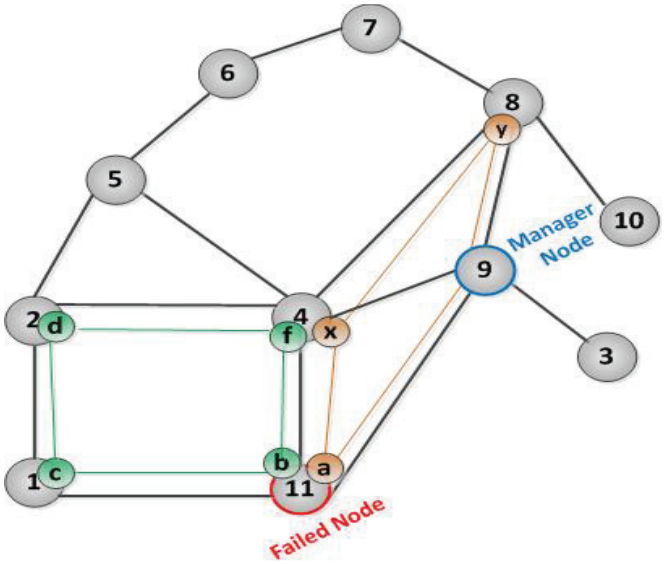

Fig. 1: Physical network with failed node that hosting two virtual nodes

The VN mapping process takes place in two steps [4] [19]: Node mapping ($\mathcal{M}_N : N_v \rightarrow N_s$) and Link mapping ($\mathcal{M}_L : L_v \rightarrow P_s$), where P_s is a set of substrate links. Each virtual link can be mapped to an unsplittable substrate path or a set of splittable substrate paths [4].

After a VN is deployed, the remaining capacity of the substrate resources is referred to the residual cpu capacity $cpu_{res}(n_s)$ of the substrate nodes and the residual bandwidth capacity $bw_{res}(l_s)$ of the substrate links, and those two variables (i.e. $cpu_{res}(n_s)$ and $bw_{res}(l_s)$) are two factors used to indicate the load of the physical network.

In our approach, we assume that the mapping of the VN requests has already been performed and the manager nodes in the physical network are assigned in the mapping phase.

Each manager node (e.g., node 9 in Fig. 1 manages node 11) maintains a table (e.g., Table I) that contains the IDs of the VNs having virtual nodes hosted on the managed substrate node. Also, the table maintains references to the hosted VN nodes in the substrate node (e.g., nodes a and b in Fig. 1), and the adjacent virtual nodes to the hosted nodes (e.g., nodes f and c in Fig. 1). This table is updated continuously (e.g., every time a VN is accepted or rejected).

Figure 1 depicts an example of a SN with two already embedded VNs. The failed node (node 11) has been assigned a manager node (node 9) that is responsible for VN nodes reallocation.

The InP's revenue received from serving a VN request can be calculated as follows:

$$R(G_v) = T(G_v)\Big(f_{cpu} \sum_{n_v \in N_v} cpu(n_v) + f_{bw} \sum_{l_v \in L_v} bw(l_v)\Big) \quad (1)$$

Where $T(G_v)$ represents the life-time of the hosted VN G_v. The price of used units of the cpu and bw capacities is denoted by f_{cpu} and f_{bw}, respectively.

The InP generates a Service Level Agreement (SLA) [7] that defines, in addition to the terms that manage the relationship between the InP and the VN owner, the monetary penalty charge that network InP has to pay in the event of service interruption due to node or link failure.

The cost of allocated substrate resources to the VN can be defined as follows:

$$C(G_v) = T(G_v)\Big(c_{cpu} \sum_{n_v \in N_v} cpu(n_v) + c_{bw} \sum_{l_v \in L_v} \sum_{l_s \in L_S} bw_{l_v}^{l_s}\Big) \quad (2)$$

Where $bw_{l_v}^{l_s}$ denotes the amount of the allocated bandwidth on the substrate link l_s for the virtual link l_v, and c_{cpu} and c_{bw} are the costs of the allocated cpu and bandwidth, respectively.

Let p denotes the monetary penalty charge incurred per a time unit when one node in VN is hosted on a failed node n_s^{failed}, or one VN link pass through the failed node n_s^{failed}. The penalty charge paid by the InP to a single VN owner in case of node failure can be calculated as the following:

$$P(G_v) = \sum t_{int}(n_s^{failed}).p \quad , \quad \exists n_v \in N_v : \mathcal{M}(n_v) = n_s^{failed} \quad or \quad \exists l_v \in L_v : bw_{l_v}^{l_s} \neq 0, l_s \quad incident \quad to \quad n_s^{failed} \quad (3)$$

Where $t_{int}(n_s^{failed})$ is the service interruption period caused by node failure n_s^{failed}.

For any node failure in the SN at a time t, the InP's profit for a single VN request (G_v) that are affected by node failure will be reduced such that:

$$Profit(G_v) = R(G_v) - C(G_v) - P(G_v) \quad (4)$$

B. Node failure recovery problem description

The main objective of the proposed work is to develop a self-recovering VN mechanism that can minimize the service interruption period and node failure recovery cost, and maintain a high level of physical network performance, which in turn increases the InP's profit (4).

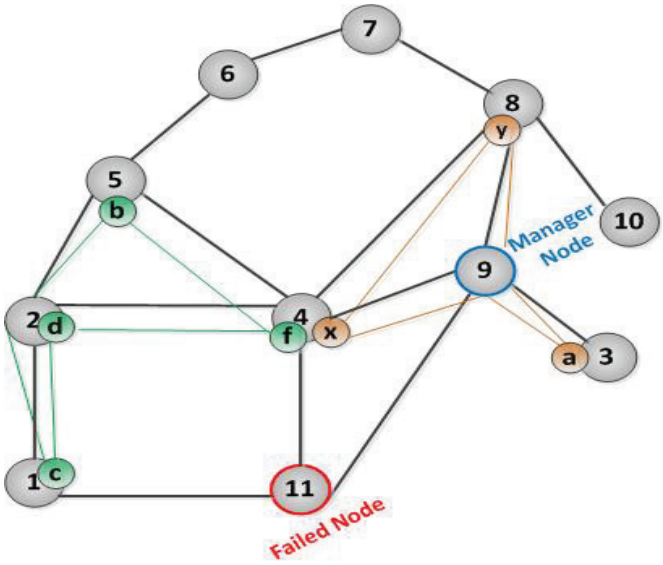

Fig. 2: The state of physical network after recovery

The mechanism would recover from a node failure without any human intervention or network provider consultation. In order to reach the objective of self-recovering VNs, we need to separate the node failure recovery procedure from the VN embedding algorithm and this is realized by distributing the relocation procedure of the affected nodes in the failed node.

The problem addressed in this paper is how to relocate the hosted virtual nodes in the failed node with the objective of minimizing the node failure recovery cost and service interruption period. Our approach emphasizes that any virtual node relocation must be done locally and coordinated only by manager nodes.

IV. PROPOSED RECOVERY ALGORITHM

The process of relocating the VN nodes hosted on the failed node n_s^{failed} is triggered when the InP sends a recovery request to the corresponding manager node (e.g., node 9 in Fig. 1). The recovery process for each affected VN node proceeds as follows: the manager node forwards the recovery request to all SN nodes hosting virtual nodes adjacent to affected VN nodes (e.g., node 1 and node 4 in Fig. 1). Each of these nodes construct a shortest path tree (SPT) to all SN nodes at a distance at most ϵ [1] hops from the node with the root of the SPT representing the node itself. The manager node employs these paths to select the node with the shortest distance to all the SN nodes that are hosting VN nodes adjacent to that on the failed node. This node eventually becomes the optimal candidate node to host the affected VN node. The length of the resulted paths from SPT must be less than or equal to ϵ, the maximum number of hops allowed for a path that maps a VN link. Also, the capacity of the end nodes [2] of the paths must be greater than or equal to the capacity of the virtual node hosted on the failed node (e.g., node b). The manager node selects the optimal candidate node by computing the intersection of the paths resulted from running SPT algorithm for all SN nodes hosting neighbour VN nodes of the affected node in failed node. The intersected paths are computed by finding the nodes that are common in the SPTs. Then, we select, from these nodes, the node with the min path cost to all the root nodes in the SPTs. The paths on the SPTs to this node are the intersected paths. The paths with minimum length and end nodes of the paths with maximum capacity are selected. Fig. 2 depicts the graph resulting from executing the node failure recovery algorithm. Algorithm 1 provides a peseudo code description of the node failure recovery algorithm.

[1]The threshold for the max number of links is decided by the SP.
[2]The term end node is used to refer to the last node in the path.

Algorithm 1 Node Failure Recovery

1: **for all** $\mathcal{M}(n_v)$=n_s^{failed} **do**
2: S_1 (set of paths) $\leftarrow \phi$
3: S_2 (set of the intersected paths) $\leftarrow \phi$
4: $N \leftarrow \{ \mathcal{M}(n_v^{'}) : \exists\ l_v(n_v, n_v^{'})$ & $\mathcal{M}(n_v)$ =n_s^{failed} $\}$
5: Manager node sends SPT request to all SN nodes in N
6: **for all** $i = 1$ to |N| **do**
7: Current SN node in N run **PathCompute**
8: **U**pdate S_1
9: **end for**
10: $S_2 \leftarrow$ compute the intersected paths from S_1
11: **S**elect intersected paths from S_2 with minimized length && Max cpu(end node) from the selected intersected paths
12: **if** selection not successful **then**
13: **R**un Reconfig
14: **end if**
15: **end for**
16: **for all** paths pass through n_s^{failed} **do**
17: Manager node runs shortest path algorithm among end nodes of the path
18: **end for**

V. NODE FAILURE RECOVERY FOR OVERLOADED NETWORKS

In the overloaded substrate network, the probability of failing to find a new optimal candidate node for the VN nodes hosted on the failed node will be high. There are two possible causes of this event: overloaded substrate nodes and bottlenecked substrate links. To operate the node failure recovery approach in the overloaded network, a reconfiguration procedure is first executed to migrate one or more virtual nodes hosted on the overloaded substrate nodes to free up some cpu capacity in the substrate node or some bandwidth in the SN links to be able to relocate the hosted VN node in failed node.

A. Insufficient CPU capacity of the nodes hosting SPTs nodes

The process of reconfiguration starts by sorting all the virtual nodes hosted on the SPT nodes. The criteria (CRT in line 1 in Algorithm 3) used for sorting these VN nodes are: the residual CPU capacity of the nodes hosting these VN nodes and the life time of the VNs containing these VN nodes. Then, the recovery procedure is executed on the first sorted VN node that carries the highest cpu capacity to be relocated to a new SN node. If the migration of the virtual node succeeds

in releasing enough capacity, then the node failure recovery mechanism is executed again to relocate the affected VN node in the failed node. Otherwise, the above procedure is repeated on the rest of the sorted list of virtual nodes until affected VN node relocation succeeds, (see Algorithm 3).

B. SPTs traverse bottlenecked SN links

When the constructed SPTs pass through bottlenecked substrate link or links, a virtual link on a bottlenecked SN link is relocated to release enough bandwidth to be used again in node failure recovery. First, all the VN nodes hosted on the SPTs nodes are sorted according to the following criteria (*CRT*, line 1 in Algorithm 3): the life time of the VN where virtual nodes belong to and the number of the overloaded SN links where SPTs pass through. Next, the recovery procedure is executed to migrate the first sorted virtual node. If it fails to release enough bandwidth, the recovery procedure is executed on the next sorted VN node. The execution of recovery procedure is repeated on the rest of the listed VN nodes until enough bandwidth become available to relocate the affected virtual node on the failed node. After enough bandwidth is released in the overloaded links, node failure recovery algorithm is executed to relocate the affected virtual node on the failed SN node, (see Algorithm 3).

Algorithm 2 PathCompute

```
1: Run SPT algorithm to all nodes in SN not hosting virtual node
   used by VN using the virtual node need to be relocated
2: pSet ← Store the resulted paths from running SPT
3: for all paths ∈ pSet do
4:    if length (current path)≤ ϵ && cpu(end node)≥ cpu capacity
      of the virtual node need to be relocated then
5:       S1 ← Add current path
6:    end if
7: end for
```

The procedure in Algorithm 2 computes the SPTs that are deployed to generate the set of intersected paths for all the neighbours of the affected VN node in the failed SN node, this procedure is used in Algorithms 1 and 3.

The failed SN node may not only be hosting some virtual nodes, but some paths from different VNs might also traverse it. Node failure recovery approach runs the shortest path algorithm over the end nodes of those paths to relocate them to pass through any node other than failed node. To get a new SN path, the manager node sends a shortest path request to any end node of the SN path hosting the VN link need to be relocated. Next, the SN node executes the shortest path algorithm to the other end node of the path, and when creating a new path, the manager node reallocates the virtual link to this new SN path.

VI. PERFORMANCE EVALUATION

In this section, we first describe the simulation environment in which we evaluate our approach, and then we present the evaluation results. The main focus of our evaluation is to demonstrate the benefits of our approach in reducing the node failure recovery cost and service interruption time over Star-shaped VN component migration scheme [5] in the event of node failure.

Algorithm 3 Reconfig

```
1: z ← Sort out all the virtual nodes decreasingly, those virtual
   nodes are hosted on the nodes over which intersected paths
   traverse, sorting is based on the criteria CRT
2: Set done = false
3: while done ≠ true do
4:    S1 (set of paths) ← ϕ
5:    S2 (set of the intersected paths) ← ϕ
6:    N ← { M(n'v) : ∃ lv(nv in z ,n'v)}
7:    for all i = 1 to |N| do
8:       Current SN node in N run PathCompute
9:       Update S1
10:   end for
11:   S2 ← compute the intersected paths from S1
12:   Select intersected paths with minimized length from S2 &&
      end node with max capacity from the selected intersected paths
13:   if selection succeed then
14:      done=true
15:   end if
16: end while
```

A. Simulation Environment

We have extended the implementation of the discrete event simulator "ViNE-Yard" [1] to evaluate the performance of the proposed node recovery approach, and to demonstrate the advantages of running node failure recovery in a decentralized manner. The GT-ITM tool [18] is used to generate the random substrate network and the VN requests.

Following previous work [4] [17] [19], we adopt the following simulation scenario: the SN topologies used in our experiments are set to 50 nodes. Each pair of nodes is randomly connected with probability 0.5. The CPU of the substrate nodes and the bandwidth of the links are real numbers uniformly distributed between 50 and 100. In the VN requests, the number of nodes is randomly determined by a uniform distribution between 2 and 15. Following [13], assume that the VN requests and substrate node failure events follow a Poisson process with arrival rates λ_V and λ_F respectively. We assume that λ_V and λ_F are 4 events per 100 time units. The VN lifetime is modeled by an exponential distribution with an average of μ =1000 time units. The CPU requirements of the nodes on the VN are real numbers uniformly distributed between 0 to 20 and the bandwidth requirements of the virtual links are uniformly distributed between 0 to 50. The failure node is selected randomly from the substrate network during our simulations.

The metrics we use to measure the proposed algorithm performance are: the experienced delay, node failure recovery cost, and success rate.

To the best of our knowledge there are no other existing node failure recovery techniques that we can compare our scheme against. Nonetheless, we found that Star-shaped VN component migration scheme [5] is the closest technique that can be used for comparison against proposed work.

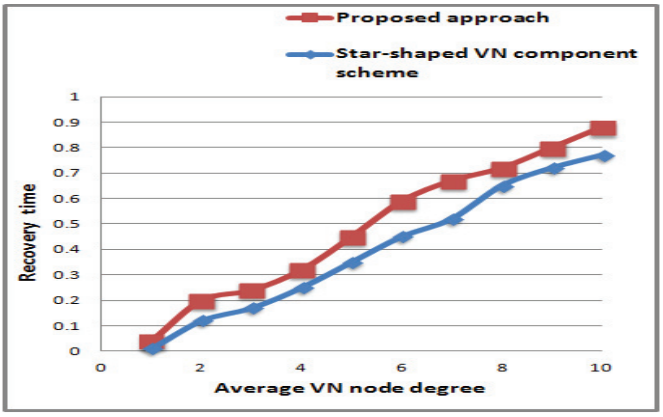

Fig. 3: Average VN node degree vs. time taken for VN node relocation

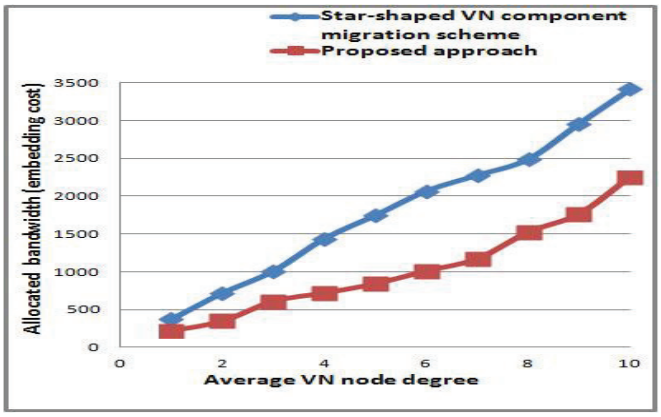

Fig. 5: Average VN node degree vs. cost (allocated bandwidth)

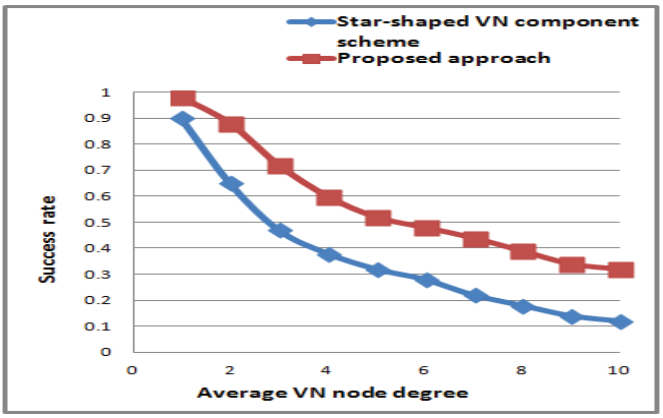

Fig. 4: Average VN node degree vs. success rate

B. Evaluation results

Several VN topologies have been used to evaluate the proposed algorithm in terms of the number of neighbors of the hosted virtual nodes in the failed node versus the time needed to relocate those hosted virtual nodes. As shown in Figure 3, the time required to relocate any hosted virtual node, in the proposed approach and Star-shaped VN component migration scheme, increases with the average virtual node degree on failed node. Clearly, all the neighbors of the hosted virtual node in the failed node need to run an SPT algorithm using the proposed approach and shortest path algorithm in Star-shaped VN component migration scheme. Running SPT and shortest path algorithms adds more time to the hosted virtual node relocation process. Furthermore, there is more time added in the proposed approach to run the reconfiguration procedure when the node relocation fails in the first try, and this will delay the relocation process. Figure 3 depicts that the time taken to relocate nodes using the proposed approach is more than Star-shaped VN component migration scheme, the reason behind this is due to the use of the SPT algorithm and reconfiguration procedure.

However, the inclusion of the reconfiguration procedure enhances the performance of the proposed approach in terms of the success rate of node failure recovery process. Figure 4 demonstrates the benefit of using the proposed approach in increasing the success rate of node failure recovery over Star-shaped VN component migration scheme. The figure shows that the success rate of the proposed approach is higher than Star-shaped VN component migration scheme, the reason is due to the existence of reconfiguration process in the proposed approach that is triggered when the node relocation is failed in the first try, and this increases the opportunity of success of node failure recovery process even when it is rejected in the first try.

Next, we compare the proposed approach against Star-shaped VN component migration technique in terms of node failure recovery cost, the amount of bandwidth allocated to the affected virtual links when moving the failed node. Figure 5 illustrates that proposed approach is less expensive than Star-shaped VN component migration scheme since the intersection criteria of the paths in the produced SPTs will be based on the minimized number of edges for each path. This will reduce the cost of embedding VNs on the SN that is illustrated in equation (2), which in turn increases the profit gained from VNs as demonstrated in equation (4).

VII. CONCLUSION AND FUTURE WORK

In this paper, we designed and implemented a novel decentralized node recovery approach for virtual networks (VNs). The main objective of the proposed algorithm is to make the virtual networks more reliable by reducing the service interruption period in the event of node failure and minimizing the node failure recovery cost. In this study, we propose a node failure recovery scheme using a decentralized manner to relocate the hosted virtual nodes in the failed node. In fact, the manager node will be leading the process of relocating the virtual nodes from the failed node without coordination with the InP. The performance evaluation results obtained show that our approach is more applicable compared to other approaches since it runs node failure recovery in reduced time and minimised cost.

REFERENCES

[1] "ViNE-Yard," http://www.mosharaf.com/ViNE-Yard.tar.gz.
[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner. *"Overcoming the Internet impasse through virtualization,"* Computer, vol. 38, pp. 34-41, April 2005.

[3] N. F. Butt, M. Chowdhury, and R. Boutaba. *"Topology-awareness and reoptimization mechanism for virtual network embedding,"* IFIP NETWORKING Conference, pp. 27-39, 2010.
[4] N.M.Mosharaf Kabir Chowdhury, Muntasir Rahman and Raouf Boutaba.*"*Virtual Network Embedding with Coordinated Node and Link Mapping," IEEE INFCOM, 2009.
[5] Ilhem Fajjari, Nadjib Aitsaadi, Guy Pujolle and Hubert Zimmermann. *" VNR Algorithm: A Greedy Approach For Virtual Networks Reconfigurations,"* IEEE Globel Communications Conference, Exhibition and Industry Forum, Houston: United States, 2011.
[6] N. Feamster, L. Gao, and J. Rexford. *"How to lease the internet in your spare time,"* ACM SIGCOMM Computer Communication Review,vol. 37, no. 1, pp. 61-64, 2007.
[7] E. Keller, R. Lee, and J. Rexford. *"Accountability in Hosted Virtual Networks,"* ACM 978-1-60558-595-6/09/08,VISA09, August 17, 2009.
[8] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. OToole, Jr.*"Overcast: Reliable multicasting with an overlay network,"* in Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI), October 2000, pp. 197212.
[9] J. Lu and J. Turner.*"Efficient mapping of virtual networks onto a shared substrate,"* Washington University, Technical Report, WUCSE-2006-35, 2006.
[10] N.M. Mosharaf and R. Boutaba. *"A Survey of Network Virtualization,"* University of Waterloo, Ontario, Canada, Technical Report CS-2008-25,October 2008.
[11] Panagiotis Papadimitriou, Olaf Maennel, Adam Greenhalgh, Anja Feldmann and Laurent Mathy,*"Implementing Network Virtualization for a Future Internet,"* 20th ITC Specialist Seminar, Hoi An, Vietnam, May 2009.
[12] Barry Porter, Francois Taiani and Geoff Coulson, *"Generalizing Repair for Overlay Networks,"*Computing Department, Lancaster University, UK,Technical Report, PTC-06-01, October 2006.
[13] Muntasir Raihan Rahman, Issam Aib, and Raouf Boutaba,*"Survivable Virtual Network Embedding,"* IFIP International Federation for Information Processing, 2010.
[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan,*"Chord: A scalable peer-to-peer lookup service for internet applications,"* in Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications. ACM Press, 2001, pp. 149160.
[15] J. Turner and D. Taylor. *"Diversifying the Internet,"* in IEEE GLOBECOM, vol. 2, 2005.
[16] Yi Wang ,Eric Keller, Brian Biskeborn , Jacobus van der Merwe and Jennifer Rexford. *" Virtual routers on the move : live router migration as a network-management primitive,"* SIGCOMM,pp. 231-242. ACM, New Yor, 2008.
[17] M. Yu, Y. Yi, J. Rexford, and M. Chiang.*"Rethinking virtual network embedding: Substrate support for path splitting and migration,"* Princeton University, Technical Report TR-788-07, July 2007.
[18] E. Zegura, K. Calvert, and S. Bhattacharjee. *"How to model an Internetwork,"* in Proceedings of IEEE INFOCOM, 1996, pp. 594-602.
[19] Y. Zhu and M. Ammar. *"Algorithms for assigning substrate network Resources to virtual network components,"* in Proc. IEEE INFOCOM, 2006.