Progressive Recovery For Network Virtualization After Large-Scale Disasters

Mahsa Pourvali[1], Kaile Liang[2], Feng Gu[3], Hao Bai[1], Khaled Shaban[4], Samee Khan[5], Nasir Ghani[1]
[1]*University of South Florida,* [2]*University of New Mexico,* [3]*VM Ware,* [4]*Qatar University,* [5]*ND State University*

***Abstract*—Network virtualization allows users to build customized interconnected storage/computing configurations for their business needs. Today this capability is being widely used to improve the scalability and reliability of cloud-based services, including virtual infrastructures services. However, as more and more business-critical applications migrate to the cloud, disaster recovery is now a major concern. Although some studies have looked at network virtualization design under such scenarios, most have only studied pre-fault protection provisioning. Indeed, there is a pressing need to address post-fault recovery, since physical infrastructure repairs will likely occur in a staged progressive manner due to constraints on available resources. Hence this paper studies progressive recovery design for network virtualization and evaluates several heuristic strategies.**

***Index Terms*—Cloud networking, network virtualization, disaster recovery, progressive recovery**

I. INTRODUCTION

Cloud-based applications are gaining strong traction with businesses and private end-users. These offerings preclude costly management overheads and are commonly hosted at external datacenter sites with abundant storage and computing resources. However, as service demands grow, there is a further need to host applications across multiple datacenters for improved responsiveness. As a result, many providers are using underlying *network virtualization* techniques [1] to build and interconnect customized distributed storage/computing configurations to support their customer needs, e.g., also termed *infrastructure as a service* (IaaS) offerings. Specifically, *virtual networks* (VN) are being provisioned to interconnect physical datacenter resources via substrate networks.

Now from a service provisioning perspective, network virtualization requires *VN embedding* schemes (VNE) to map demands over physical substrates. For example, consider a VN request comprising of several VN nodes (computing, storage) and VN links (bandwidth capacity). Here, the VN nodes must be mapped to resource pools at unique substrate datacenter sites, and similarly VN links must be mapped to bandwidth connections. Along these lines, a range of VNE algorithms have been developed to achieve objectives such as revenue maximization, cost reduction, etc [1].

However, as more and more users migrate to the cloud, survivability and availability concerns are coming to the fore. For example, datacenter outages or network link failures can easily disrupt many VN mappings, causing service degradation and even *service level agreement* (SLA) violations. These issues are particularly acute for large-scale stressor events causing multiple outages, e.g., natural disasters, malicious *weapons of mass destruction* (WMD) attacks, cascading power outages, etc [2],[3]. Hence a range of VN survivability schemes have been developed for single link failures [4],[5], single node failures [6],[7], and even disaster recovery [8]-[10]. In particular, the latter solutions are designed for "mission-critical" applications which require very high levels of service availability. For the most part, the above strategies focus on protection methodologies to pre-provision backup resources before a failure(s). Hence it is clearly very difficult, if not impossible, to guarantee recovery under general disaster conditions with multiple random/correlated failures [2],[3].

In light of the above, there is a growing need to address *post-fault* VN recovery concerns for cloud services. Namely, infrastructure repairs will likely occur in multiple *stages* as backup resources are brought to the field and installed, i.e., routers, switches, links, transceivers, regenerators, etc. However, damaged substrate infrastructures will still have to operate in a degraded manner and provide some partial level of service for clients. This transition is commonly referred to as *progressive* recovery [11] and terminates when infrastructures are fully restored. Clearly the sequencing of datacenter node and network link recoveries will have a direct impact on VN services, i.e., scheduling of repair resources. However, most studies have only looked at progressive recovery for point-to-point connections and not complex VN demands [11],[12].

Along these lines, this paper presents novel progressive recovery schemes for VN services. Section II first presents a background survey of VN survivability and progressive recovery methods. Novel VN progressive recovery schemes are then detailed in Section III based upon a variety of heuristic methods. Detailed performance evaluation results are then presented in Section IV, along with conculsions and future research directions in Section V.

II. BACKGROUND

The VNE problem is NP-hard and a variety of solutions have been developed, see survey in [1]. These solutions include complex optimization-based strategies to minimize resource usages or maximize revenues, as well as more tractable heuristic algorithms (single-stage, two-stage). Given the increased focus on service reliability, recent studies have also looked at VN survivability design. Of particular concern are large-scale disaster events which can cause extensive damage to physical facilities, e.g., natural disasters, cascading power outages, malicious WMD attacks, etc [3].

For example, several studies have looked at VN recovery from single link failures, e.g., by computing link-disjoint path-pairs for each embedded VN link connection, see studies in

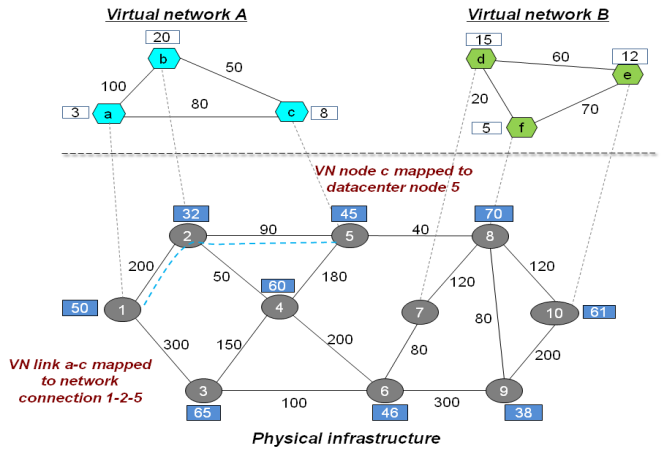

Fig. 1. Physical substrate (network, datacenter) w. mapped VN demands

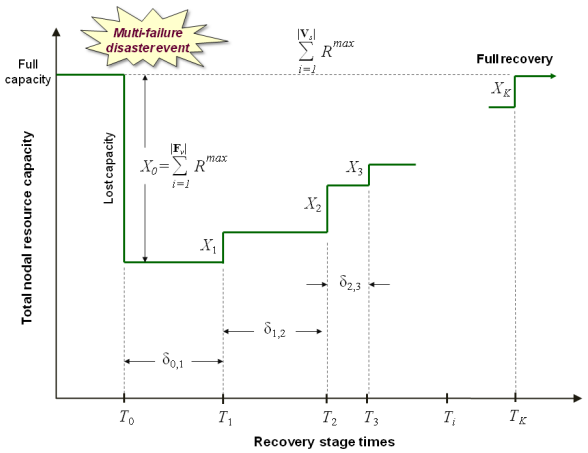

Fig. 2. Progressive recovery stages (node resources shown)

[4],[5]. Meanwhile, others have proposed single node failure recovery schemes by provisioning one or more backup nodes, see [6],[7]. More recently, several efforts have also studied VN disaster recovery for multiple correlated node/link failures. These solutions assume pre-defined risk (stressor) regions with given occurrence probabilities and conditional node and link failure rates, as per models in [2]. Using these inputs, various VN protection strategies are developed to pre-provision backup VN nodes/links. For example, the schemes in [8] and [9] compute independent VN mappings for each risk region and condense them into a single mapping by using resource sharing on common nodes and links. An incremental scheme is also proposed by adding backup VN nodes/links to protect against different stressor regions. Meanwhile, [10] proposes more efficient solutions using region partitioning techniques to compute disjoint primary/backup mappings.

However, the above solutions only focus on backup resource pre-provisioning *prior* to fault events. Hence it is very difficult and costly to pre-protect against all possible multi-failure combinations here. In other words, VN services can still fail and will require further *physical* infrastructure repairs. Now such recovery will likely occur in multiple stages as operators deploy backup resources. Hence there is a pressing need to develop progressive post-fault recovery schemes to judiciously schedule repair resources. To date, however, only a handful of studies have looked at progressive network recovery for point-to-point connection demands/flows. For example [11] and [12] present several related optimization-based strategies along with approximation solutions.

III. PROGRESSIVE RECOVERY SCHEMES

To address the above concerns, some novel VN progressive recovery schemes are now presented. The goal here is to intelligently place node and link repair resources to rapidly recover failed VN loads and hence reduce service downtimes/penalties. The requisite notation is first introduced.

The underlying *physical* substrate is represented by an undirected graph $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$), where $\mathbf{V}_s$ is the set of datacenter nodes/sites and $\mathbf{E}_s$ is the set of network links. Each datacenter node $v_i \in \mathbf{V}_s$ has maximum resource capacity of R^{max} units. Similarly, each physical substrate link $e_i \in \mathbf{E}_s$ has maximum bandwidth capacity of B^{max} units. A substrate link e_i is also denoted by (v_i, v_i'), where v_i and v_i' are the two link endpoints. Meanwhile, a VN request is given by an undirected graph, $\mathbf{G}_v$=($\mathbf{V}_v, \mathbf{E}_v$), where $\mathbf{V}_v$ is the set of virtual nodes and $\mathbf{E}_v$ is the set of virtual links. Here each VN node $v_i \in \mathbf{V}_v$ needs r_i in node resources and each VN link $e_i \in \mathbf{E}_v$ needs b_i in bandwidth capacity. Akin to physical substrate links, a VN link e_i is also denoted by (v_i, v_i'), i.e., by its two *virtual* node endpoints. Figure 1 shows a sample 10-node substrate network hosting two VN requests (numbers represent resources).

Next, consider the notation for progressive recovery. An initial multi-failure stressor event occurs at T_0 and is followed by a series of recovery stages at times T_j, $j = 1, 2, ...$ ($T_{j+1} > T_j$), Figure 2. During these recovery stages, all failed nodes and links will either transition directly from an initial failed state to a fully-recovered state or through a partially-recovered state. Hence without loss of generality, it is assumed that all damaged nodes and links after a disaster event fail completely, i.e., available resource levels fall to zero. As repair resources are subsequently installed, these resource levels start to rise and eventually recover to their pre-fault maximums by the last recovery stage, i.e., R^{max} or B^{max} units.

Based upon the above, all failed and partially-recovered nodes and links are termed as *affected* and deemed eligible to receive repair resources. Namely, the sets of affected nodes and links in stage j are denoted by $\mathbf{F}_v^j \subset \mathbf{F}_v^{j-1}$ and $\mathbf{F}_e^j \subset \mathbf{F}_e^{j-1}$, respectively. Here, index j=0 denotes the initial stressor after which the aggregate datacenter node resources drop by $X_0 = \sum_{v_i \in \mathbf{F}_v^0} R^{max}$ units and the aggregate network link capacity drops by $Y_0 = \sum_{e_i \in \mathbf{F}_e^0} B^{max}$ units. Meanwhile, the amount of datacenter repair resources received in the j-th recovery stage is given by X_j, e.g., computing racks, storage disks. Similarly, the amount of link repair resources received is given by Y_j, e.g., optical link transponders, switching units. Hence full recovery implies that $X_0 = \sum_{j=1}^{k} X_j = |\mathbf{F}_v^0| R^{max}$ and $Y_0 = \sum_{j=1}^{k} Y_j = |\mathbf{F}_e^0| B^{max}$ over k stages.

The progressive recovery schemes first distribute available

Given failed datacenter node/network link sets ($\mathbf{F}_v$, $\mathbf{F}_e$) and datacenter node/network link repair resources (X_j, Y_j units, respectively)
Initialize variables $x = X_j$, $y = Y_j$
Define eligible node subset, $\mathbf{F}_v^{'j}$, where $\forall v_i \in \mathbf{F}_v^{'j}$, at least one neighbor of v_i in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$) is non-failed
Define eligible links subset, $\mathbf{F}_e^{'j}$, where $\forall e_i = (v_i, v_i') \in \mathbf{F}_e^{'j}$, v_i, v_i' not fully-failed
Compute increments Δ_x^{uni}, Δ_y^{uni} (Eqs. 1,2)
for i=1 to $|\mathbf{F}_v^{'j}|$
    $R_i = R_i + min(R^{max} - R_i, \Delta_x^{uni})$, v_i is i-th entry in $F_v^{'j}$ and R_i is its current resource level
    if $(R_i = R^{max})$
        $\mathbf{F}_v^j \rightarrow \mathbf{F}_v^j - \{v_i\}$
for i=1 to $|\mathbf{F}_e^{'j}|$
    $B_i = B_i + min(B^{max} - B_i, \Delta_y^{uni})$, e_i is i-th entry in $F_e^{'j}$ and B_i is its current resource level
    if $(B_i = B_{max})$
        $\mathbf{F}_e^j \rightarrow \mathbf{F}_e^j - \{e_i\}$

Fig. 3. Uniform placement (UP) algorithm, j-th recovery stage

repair resources among failed nodes/links and then use any standard VNE algorithm to re-map failed demands. The main objective here is to allocate repair resources to minimize disruption, i.e., maximize the number of restored VN requests. To ensure that repair resources can be accessed right away or by the next stage, all schemes only place resources at eligible nodes which have at least one non-failed neighbor node, i.e. fully or partially-recovered nodes. Similarly, link repair resources are only placed at failed or partially-recovered physical links with *non-failed* endpoints, i.e., working or partially-recovered endpoint nodes. This selectivity helps limit islanding and improves network connectivity during recovery. Therefore the set of affected nodes in stage j, $\mathbf{F}_v^j$, is further partitioned into a *candidate node* subset, $\mathbf{F}_v^{'j}$, where $\forall v_i \in \mathbf{F}_v^{'j}$, at least one *physical* neighbor of v_i in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{V}_s$) is not fully-failed. Similarly, the set of affected links, $\mathbf{F}_e^j$, is also partitioned into a *candidate link* subset, $\mathbf{F}_e^{'j}$, where $\forall e_i = (v_i, v_i') \in \mathbf{F}_e^{'j}$, both v_i, v_i' in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{V}_s$) are not fully-failed. Consider the details.

A. Uniform Placement (UP)

This scheme distributes repair resources in an even manner across damaged nodes and links, see Figure 3. Namely, each candidate datacenter node receives an even portion of the X_j node repair resources in stage j:

$$\Delta_x^{uni} = \left\lfloor \frac{X_j}{|\mathbf{F}_v^{'j}|} \right\rfloor \tag{1}$$

Furthermore, each eligible candidate link also receives an even portion of the Y_i link repair resources in stage j:

$$\Delta_y^{uni} = \left\lfloor \frac{Y_j}{|\mathbf{F}_e^{'j}|} \right\rfloor \tag{2}$$

Given eligible datacenter node/network link sets ($\mathbf{F}_v$, $\mathbf{F}_e$) and datacenter node/network link repair resources (X_j, Y_j units, respectively)
Initialize variables $x = X_j$, $y = Y_j$, $done = 0$
while $(!done)$
    Define candidate node subset, $\mathbf{F}_v^{'j}$, where $\forall v_i \in \mathbf{F}_v^{'j}$, at least one neighbor of v_i in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$) is non-failed
    Define eligible links subset, $\mathbf{F}_e^{'j}$, where $\forall e_i = (v_i, v_i') \in \mathbf{F}_e^{'j}$, v_i, v_i' not fully-failed
    Randomly select node from $\mathbf{F}_v^{'j}$, v_i
    $\Delta_x^{rnd} = min(R^{max} - R_i, x)$, where R_i is node v_i current resource level
    Add capacity to v_i, $R_i = R_i + \Delta_x^{rnd}$
    $x = x - \Delta_x^{rnd}$
    if $(R_i = R^{max})$
        $\mathbf{F}_v^j \rightarrow \mathbf{F}_v^j - \{v_i\}$
    Randomly select link from $\mathbf{F}_e^{'j}$, e_i
    $\Delta_y^{rnd} = min(B^{max} - B_i, y)$, where B_i is link e_i current resource level
    Add bandwidth to e_i, $B_i = B_i + \Delta_y^{rnd}$
    $y = y - \Delta_y^{rnd}$
    if $(B_i = B^{max})$
        $\mathbf{F}_e^j \rightarrow \mathbf{F}_e^j - \{e_i\}$
    if $(\mathbf{F}_v^j, \mathbf{F}_e^j = \{\emptyset\}$ or $x, y = 0)$
        $done$

Fig. 4. Random placement (RP) algorithm, j-th recovery stage

B. Random Placement (RP)

This scheme assigns repair resources to failed physical datacenter nodes and network links in a random manner, Figure 4. Namely the scheme simply chooses a random candidate node $v_i \in \mathbf{F}_v^{'j}$ to receive the X_j units of incoming node repair resources. A random candidate link, $e_i \in \mathbf{F}_e^{'j}$, is then selected to receive the Y_j units of incoming link repair resources.

Now if the assigned repair resources exceed the amount needed to fully recover the selected node (link), then the surplus resources are further assigned (iteratively) to other randomly-chosen nodes (links) in $\mathbf{F}_v^{'j}$ ($\mathbf{F}_e^{'j}$). In addition, all nodes (links) recovering to their maximum pre-fault resource levels are removed from the failed sets $\mathbf{F}_v^j$ ($\mathbf{F}_e^j$). This procedure is implemented via the while loop in Figure 4 and uses appropriate tracking variables to record the leftover amounts of available node and link repair resources, i.e., x and y, respectively. Hence at each iteration, the node resource increment, Δ_x^{rnd}, is given by:

$$\Delta_x^{rnd} = min(R^{max} - R_i, x) \tag{3}$$

where R_i is the resource level at the i-th node. Meanwhle, the link resource increment, Δ_y^{rnd}, is given by:

$$\Delta_y^{rnd} = min(B^{max} - B_i, y) \tag{4}$$

where B_i is the resource level at the i-th link. This approach does not pursue any specific objectives and provides a baseline.

1: Given failed datacenter node/network link sets ($\mathbf{F}_v$, $\mathbf{F}_e$) and datacenter node/network link repair resources (X_j, Y_j units respectively)
2: Initialize variables $x = X_j$, $y = Y_j$, $done = 0$
3: **while** $(!done)$
4: Define candidate node subset, $\mathbf{F}_v^{'j}$, where $\forall v_i \in \mathbf{F}_v^{'j}$, at least one neighbor of v_i in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$) is non-failed
5: Define eligible links subset, $\mathbf{F}_e^{'j}$, where $\forall e_i = (v_i, v_i') \in \mathbf{F}_e^{'j}$, v_i, v_i' not fully-failed
6: Sort nodes in $\mathbf{F}_v^{ji}$ and links in $\mathbf{F}_e^{'j}$ in descending order of node degree and endpoints node degree in original physical graph $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$), respectively
7: Select first node in $\mathbf{F}_v^{'}$, v_i
8: $\Delta_x^{pnld} = min(R^{max} - R_i, x)$, , where R_i is node v_i current resource level
9: Add capacity to v_i, $R_i = R_i + \Delta_x^{pnld}$
10: $x = x - \Delta_x^{pnld}$
11: **if** $(R_i = R^{max})$
12: $\mathbf{F}_v^j \rightarrow \mathbf{F}_v^j - \{v_i\}$
13: Select first link $\in \mathbf{F}_e^{'j}$, e_i
14: $\Delta_y^{pnld} = min(B^{max} - B_i, y)$, , where B_i is link e_i current resource level
15: Add bandwidth to e_i, $B_i = B_i + \Delta_y^{pnld}$
16: $y = y - \Delta_y^{pnld}$
17: **if** $(B_i = B^{max})$
18: $\mathbf{F}_e^j \rightarrow \mathbf{F}_e^j - \{e_i\}$
19: **if** $(\mathbf{F}_v^j, \mathbf{F}_e^j = \{\emptyset\}$ or $x, y = 0)$
20: $done$

Fig. 5. Physical node and link degree (P-NLD) algorithm, j-th recovery stage

1: Given eligible datacenter node/network link sets ($\mathbf{F}_v^j$, $\mathbf{F}_e^j$) and datacenter node/network link repair resources (X_j, Y_j units respectively)
2: Initialize variables $x = X_j$, $y = Y_j$, $done = 0$
3: **while** $(!done)$
4: Define candidate node subset, $\mathbf{F}_v^{'j}$, where $\forall v_i \in \mathbf{F}_v^{'j}$, at least one neighbor of v_i in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$) is non-failed
5: Define eligible links subset, $\mathbf{F}_e^{'j}$, where $\forall e_i = (v_i, v_i') \in \mathbf{F}_e^{'j}$, v_i, v_i' not fully-failed
6: Sort $\mathbf{F}_v^{'j}$ and $\mathbf{F}_e^{'j}$ in descending order of embedded VN nodes and VN links in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$) before failure, respectively
7: Select first node in $\mathbf{F}_v^{'j}$, v_i
8: $\Delta_x^{vnld} = min(R^{max} - R_i, x)$, where R_i is node v_i current resource level
9: Add capacity to v_i, $R_i = R_i + \Delta_x^{vnld}$
10: $x = x - \Delta_x^{vnld}$
11: **if** $(R_i = R^{max})$
12: $\mathbf{F}_v^j \rightarrow \mathbf{F}_v^j - \{v_i\}$
13: Select first link $\in \mathbf{F}_e^{'j}$, e_i
14: $\Delta_y^{vnld} = min(B^{max} - B_i, y)$, where B_i is link e_i current resource level
15: Add bandwidth to e_i, $B_i = B_i + \Delta_y^{vnld}$
16: $y = y - \Delta_y^{vnld}$
17: **if** $(B_i = B^{max})$
18: $\mathbf{F}_e^j \rightarrow \mathbf{F}_e^j - \{e_i\}$
19: **if** $(\mathbf{F}_v^j, \mathbf{F}_e^j = \{\emptyset\}$ or $x, y = 0)$
20: $done$

Fig. 6. Virtual Node and link degree (V-NLD) algorithm, j-th recovery stage

C. Physical Node and Link Degree (P-NLD)

This scheme assigns repair resources to failed nodes with the highest number of physical links prior to failure, i.e., node degree in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$), see Figure 5. Similarly, link resources are assigned to links with the most connectivity before failure, i.e., endpoints with the highest node degree. The aim here is to place resources at datacenter nodes and network links that are more likely to support more VN nodes and/or VN links. As a result, the pseudocode listing for this scheme is very similar to the RP scheme, with the exception that affected nodes (links) with higher node degrees (endpoint nodes degrees) in $\mathbf{G}_s$=($\mathbf{V}_s, \mathbf{E}_s$) are selected first, Figure 5. Overall, the P-NLD scheme implements resource placement based upon static graph connectivity.

D. Virtual Node and Link Degree (V-NLD)

This scheme assigns repair resources to failed nodes (links) with higher numbers of embedded virtual nodes (virtual links) prior to the stressor event, see Figure 6. The goal here is to place resources at physical entities carrying more load in order to accelerate VN recovery. Therefore the candidate node and link sets ($\mathbf{F}_v^{'j}$, $\mathbf{F}_e^{'j}$) are first sorted in descending order of their carried virtual node/link loads, as measured immediately prior to the failure event at time T_0. The most loaded entities are then selected first, see Figure 6.

IV. PERFORMANCE EVALUATION

The progressive recovery schemes are tested using custom-developed $OPNETModeler^{TM}$ suites. Tests are done for a 24 node/86 link topology, Figure 7, where all nodes have 100 units of capacity and all substrate links have 10,000 units of bandwidth. Meanwhile, VN requests are specified as random graphs with an average of 3-5 VN nodes and 4-7 VN links each. VN nodes request between 10-20 units of capacity and VN links request between 50-1,000 units of bandwidth (uniform). All requests arrive in a random exponential manner and have infinite durations, chosen to reflect long-standing services. Also, the average amounts of node and link repair resources in each stage ($\hat{X}$, $\hat{Y}$) are set to 200 and 30,000 units, respectively. Partial VN remapping is also done using the single-stage VNE restoration algorithm in [10], albeit any other VNE scheme can be used. Finally, successive recovery stages are triggered after an average of 50 time units.

Tests are done for a disaster region failing 20% of substrate nodes/links, see Figure 7, and the results are averaged over 10 independent runs. First, the percentage of *fully-recovered* VN demands are plotted in Figure 8, i.e., restoration rate.

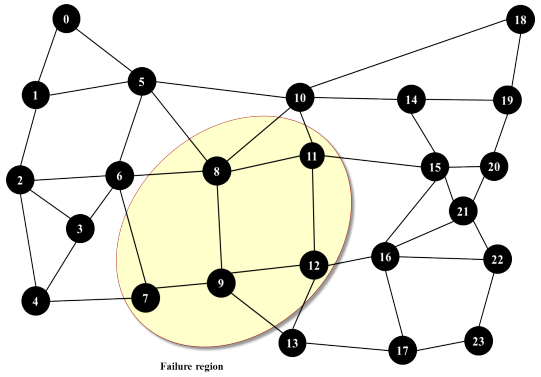

Fig. 7. Cloud datacenter and network substrate topology

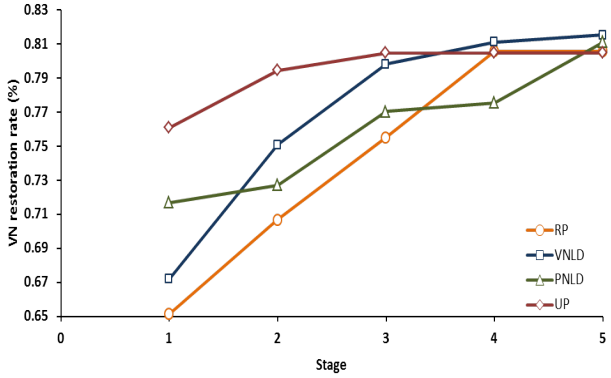

Fig. 8. VN restoration rate

Here all four schemes recover the physical network within 5 stages. Furthermore, as expected no scheme can provide full (100%) recovery of failed demands, since VN re-mapping over partially recovered substrates generally yields alternate, less efficient mappings versus the original embeddings. However, the uniform and virtual/physical degree-based schemes tend to give faster VN restoration as compared to the baseline random scheme in almost all stages. Furthermore, even though the uniform (UP) scheme provides faster initial recovery, the load-based virtual degree-based (V-NLD) scheme catches up by stage 3 and eventually gives a slightly higher final restoration rate. By contrast, the physical degree-based placement (P-NLD) scheme does not necessarily outperform the baseline random placement scheme in all stages.

Next, average VN link path lengths are plotted in Figure 9 to gauge overall bandwidth utilization (stage 0 represents the values right before the initial disaster event). These results indicate a notable and continual increase in post-fault hop count values across all schemes, i.e., almost 50% higher by final recovery stage. Furthermore, the uniform (UP) and physical degree-based (P-NLD) schemes give slightly lower usages, i.e., 5%-10% below the RP baseline. Namely, even though the basic RP scheme restores fewer failed demands (Figure 8) it does not show reduction in bandwidth usage.

V. CONCLUSIONS & FUTURE WORK

This paper studies progressive recovery for network virtualization (infrastructure) services. Several heuristic schemes

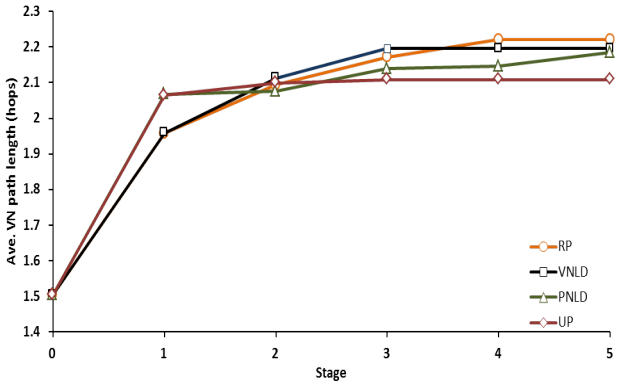

Fig. 9. Average VN path length

are outlined using uniform, random, physical node degree, and virtual node degree (load-based) resource distribution. Results indicate that uniform resource placement gives faster initial recovery of disrupted virtual networks. However, the virtual node degree scheme eventually achieves equivalent or higher recovery. On the other hand uniform and physical node degree placements yield slightly more efficient usage of network bandwidth resources. This contribution provides a good basis from which to expand in to more detailed studies using optimization and meta-heuristic strategies.

ACKNOWLEDGMENT

This work was made possible by the NPRP 5-137-2-045 grant from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

REFERENCES

[1] A. Fischer, J. Botero, *et al*, "Virtual Network Embedding: A Survey", *IEEE Comm. Surveys & Tutorials*, Vol. 15, No. 4, 4th Quater 2013.
[2] H. Lee, K. Lee, E. Modiano,"Diverse Routing in Networks with Probabilistic Failures, *IEEE/ACM Transactions on Networking*, Vol. 18, No. 6, December 2010, pp. 1895-1907.
[3] M. Rahnamay-Naeini, J. Pezoa, G. Azar, N. Ghani, M. Hayat, "Modeling Stochastic Correlated Failures and Their Effects on Network Reliability," *IEEE ICCCN 2011*, Maui, Hawaii, August 2011.
[4] Y. Chen, *et al*, "Resilient Virtual Network Service Provision in Network Virtualization Environments," *IEEE ICPADS 2010*, China, Dec. 2010.
[5] T. Guo, *et al*, "Shared Backup Network Provision for Virtual Network Embedding," *IEEE ICC 2011*, Japan, June 2011.
[6] H. Yu, *et al*, "Cost Efficient Design of Survivable Virtual Infrastructure to Recover from Facility Node Failures," *IEEE ICC 2011*, June 2011.
[7] C. Qiao, *et al*, "A novel two-step approach to surviving facility failures," *OFC 2011*, Los Angeles, CA, March 2011.
[8] H. Yu, *et al*, "Survivable Virtual Infrastructure Mapping in a Federated Computing and Networking System under Single Regional Failures," *IEEE GLOBECOM 2010*, Miami, USA, December 2010.
[9] G. Sun, *et al*, "Efficient Algorithms for Survivable Virtual Network Embedding," *Asia Communications and Photonics Conference (ACP) 2010*, China, December 2010.
[10] F. Gu, H. Alazemi, A. Rayes, N. Ghani, "Survivable Cloud Networking Services," *IEEE ICNC 2013*, San Diego, January 2013.
[11] J. Wang, C. Qiao, H. Yu, "On Progressive Network Recovery After a Major Disruption," *IEEE INFOCOM*, Shanghai, China, April 2011.
[12] S. Nurre, *et al*, "Restoring Infrastructure Systems: An Integrated Network Design and Scheduling (INDS) Approach", *European Journal of Operational Research*, 2012, pp. 794-806.