Algorithms for Assigning Substrate Network Resources to Virtual Network Components

Yong Zhu and Mostafa Ammar
Networking and Telecommunications Group
Georgia Institute of Technology, Atlanta, Georgia
Email: {yongzhu,ammar}@cc.gatech.edu

Abstract— Recent proposals for network virtualization provide a promising way to overcome the Internet ossification. The key idea of network virtualization is to build a diversified Internet to support a variety of network services and architectures through a shared substrate. A major challenge in network virtualization is to assign substrate resources to virtual networks (VN) *efficiently* and *on-demand*. This paper focuses on two versions of the VN assignment problem: VN assignment without reconfiguration (*VNA-I*) and VN assignment with reconfiguration (*VNA-II*). For the VNA-I problem, we develop a basic scheme as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are then presented to further improve the performance. For the VNA-II problem, we develop a selective VN reconfiguration scheme that prioritizes the reconfiguration of the most critical VNs. Extensive simulation experiments demonstrate that the proposed algorithms can achieve good performance under a wide range of network conditions.

Fig. 1. Network virtualization: Building a diversified Internet

I. INTRODUCTION

The Internet has been a great success in the past few decades and has provided a whole new way to access and exchange information. Its success has stimulated enormous growth and wide deployment of network technology and applications. However, the growth and deployment itself is now creating obstacles to future innovations. Specifically, due to the multi-provider nature of the Internet, adopting a new network architecture requires not only changes in individual routers and hosts, but also joint agreements among ISPs. The size and scale of today's Internet make the introduction and deployment of new network technology difficult [1], [2].

Network virtualization provides a promising way to address the ossification of the Internet [1]. In such an approach, a substrate network provider (SNP) provides a common substrate to support a number of diversified virtual networks (DVN). These DVNs, implemented by *virtual routers* connected through *virtual links*, in turn would provide a range of different communication services to applications, using a variety of protocols (shown in Figure 1). In doing so, the burdens of providing an "everything-to-everybody" network architecture or requiring universal agreements on architecture and protocol changes are released. Furthermore, the diversified Internet would make it possible for an individual or organization to rapidly deploy a DVN with potentially global scope, without making any direct investment in physical infrastructure [2].

One of the fundamental functions of network virtualization is the mapping or assigning of substrate network resources to individual vir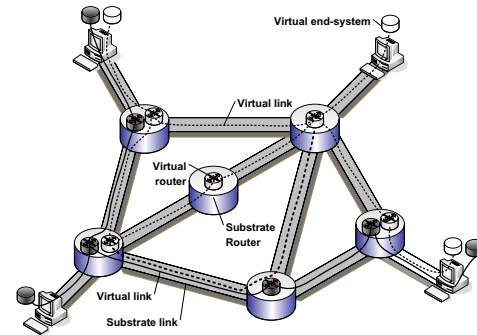tual nodes and links. Specifically, each virtual node is assigned to a substrate node and each virtual link is assigned to a substrate path that connects the corresponding end nodes. In our envisioned scenario, DVN demands would arrive at different time instances and request to set up virtual networks (VNs) with different topologies and lifetimes. Allowing VNs to be assigned to the substrate network *efficiently* and *on-demand* is desirable for the following reasons:

- Increasing efficiency in the substrate resource utilization would allow the substrate network to accommodate more VNs with limited resources and reduce hot spots or congestion.
- On-demand assignment means that the assignment for each VN is determined based on the current network situation as the demand arrives. Given that DVN demands can arrive at any moment and the SNP would not have information regarding future arrivals, it is important for the SNP to be able to make the assignment decision in response to each individual demand as they arrive.

In this work, we are motivated by the above considerations to study the following on-demand VN assignment problem: Upon the arrival of a VN request, assign its topology to the substrate network to achieve *low* and *balanced* load on both substrate nodes and links. A special case of this problem can be formulated as an unsplittable flow problem which is NP-hard. Therefore, the VN assignment problem is intrinsically difficult and heuristics will be used to solve the problem.

This paper focuses on the algorithm design for two versions of the VN assignment problem: VN assignment without reconfiguration (*VNA-I*) and VN assignment with reconfiguration (*VNA-II*). For the VNA-I problem, where the VN assignment

is fixed throughout the VN lifetime, we develop a basic scheme to achieve near optimal substrate node performance and use it as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are then presented to further improve the performance. For the VNA-II problem, we develop a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs. In doing so, we can achieve most performance benefits of the reconfiguration without excessively high cost.

The rest of this paper is organized as follows. Section II discusses the related work. Section III presents the network model and formalizes the VN assignment problem. Section IV presents algorithms for VN assignment without reconfiguration. Dynamic VN reconfiguration algorithm is given in Section V. Section VI evaluates the proposed VN assignment algorithms on a wide range of network settings and the paper concludes in Section VII.

II. RELATED WORK

The VN assignment problem shares similarities with the virtual circuit routing and virtual private network (VPN) design problems [3]–[6]. All of them involve finding paths for source/destination pairs. In both the routing and VPN design problems, locations of source/destination pairs that need to be connected are given as input parameters. In network virtualization, however, the mapping between the VN and the substrate network could be arbitrary, this extra degree of freedom increases the complexity of the problem. Furthermore, in the VPN design as well as the static routing problem, only link utilizations are considered while the VN assignment problem considers load balancing of both substrate nodes and links.

Another unique feature of the VN assignment problem is that requests arrive in terms of a graph, as opposed to a source/destination pair for the routing problem. Therefore, the VN assignment problem has both online-routing and off-line routing features: within the same VN request, multiple requests for virtual links arrive at the same time (similar to off-line routing), while no information is available for future VN requests (similar to on-line routing). This batch-arrival process allows us to consider multiple concurrent virtual link requests together to use the network resources more efficiently.

The load balancing problem on networks is a generalization of the load balancing problem on unrelated parallel machines [7]. A competitive strategy to minimize congestions in online virtual circuit routing was developed by Aspnes *et al.* to achieve a competitive ratio of $O(\log n)$ for permanent (*i.e.,* infinite holding time) virtual circuits, where n is the number of nodes in the network [8]. It was extended to the case of finite holding time circuits in [9].

The node selection and placement problem has also been studied in the context of web server replication and access network design [10]–[12]. Shi *et al.* investigated the server placement problem in overlay networks to ensure desired service quality to all customers [13]. The VN assignment problem considered in this paper also includes selecting substrate nodes for VN requests. However, the node selection is only a part

of the problem and our goal is to map the VN topology to the substrate topology to efficiently use network resources.

III. NETWORK MODEL AND PROBLEM DESCRIPTION

A. Network model

We model the topology of the substrate network as a graph $G_S = (V_S, E_S)$, where V_S is the set of substrate nodes and E_S is the set of substrate links. The i'th VN arrives at time a_i and lasts for a certain lifetime. Its topology is modelled as a graph $G_V^i = (V_V^i, E_V^i)$ with the set of virtual nodes V_V^i and the set of virtual links E_V^i respectively [1]. The assignment of the i'th VN includes following two components:

- Node assignment: Each virtual node is assigned to a different substrate node. It is formalized as a mapping $f_N^i : V_V^i \rightarrow V_S$ from virtual nodes to substrate nodes such that

$$f_N^i(\hat{v}) \in V_S, \forall \hat{v} \in V_V^i$$
$$f_N^i(\hat{u}) = f_N^i(\hat{v}), iff\, \hat{u} = \hat{v}$$

- Link assignment: Each virtual link is assigned to a substrate path between the corresponding substrate nodes. It is formalized as a mapping $f_L^i : E_V^i \rightarrow \mathcal{P}_S$ from virtual links to substrate paths such that

$$f_L^i(\widehat{\hat{u}\hat{v}}) \in P_S(f_N^i(\hat{u}), f_N^i(\hat{v})), \forall \widehat{\hat{u}\hat{v}} \in E_V^i$$

where $\mathcal{P}_S$ is the set of all substrate paths, $P_S(s,t)$ is the set of substrate paths from node s to node t. $\widehat{\hat{u}\hat{v}}$ is the virtual link from virtual node $\hat{u}$ to $\hat{v}$.

When a VN arrives, the substrate network determines the above assignments and allocates resources on the selected substrate nodes and paths accordingly. The allocated substrate resources are released when the VN departs. Furthermore, if an existing VN changes its assignment, the substrate network will release previously allocated resources by the old assignment and allocate new resources based on the new assignment.

To quantify the resource usage of the substrate network, we use the notion of *stress*. The substrate node stress at time t, $S_N(t,v)$, is defined as the number of virtual nodes that are assigned to the substrate node $v \in V_S$. Similarly, the substrate link stress at time t, $S_L(t,e)$, is defined as the number of virtual links whose substrate path passes through the substrate link $e \in E_S$. Therefore, the arrival, departure and reconfiguration of the i'th VN determines the network state transition as follows:

- The i'th VN's arrival:
$S_N(t,v) = S_N(t^-,v) + 1, \forall v \in \{f_N^i(\hat{v})|\hat{v} \in V_V^i\}$
$S_L(t,e) = S_L(t^-,e) + C(e), \forall e \in \{f_L^i(\hat{e})|\hat{e} \in E_V^i\}$

- The i'th VN's departure:
$S_N(t,v) = S_N(t^-,v) - 1, \forall v \in \{f_N^i(\hat{v})|\hat{v} \in V_V^i\}$
$S_L(t,e) = S_L(t^-,e) - C(e), \forall e \in \{f_L^i(\hat{e})|\hat{e} \in E_V^i\}$

[1]More generally, a set of mapping constraints (such as the substrate capacity of the substrate nodes/links, locations of access routers) may be attached to the VN assignment problem. In this paper, we focus on the basic scenario, where there are no constraints attached. The solution of the basic problem can be easily extended to the constrained version by checking the constraints and considering only feasible assignments.

- The i'th VN's reconfiguration:
$S_N(t,v) = S_N(t^-, v) - 1, \forall v \in \{f_N^i(\hat{v}) | \hat{v} \in V_V^i\}$
$S_N(t,v') = S_N(t^-, v') + 1, \forall v' \in \{\hat{f}_N^i(\hat{v}) | \hat{v} \in V_V^i\}$
$S_L(t,e) = S_L(t^-, e) - C(e), \forall e \in \{f_L^i(\hat{e}) | \hat{e} \in E_V^i\}$
$S_L(t,e) = S_L(t^-, e') + C(e'), \forall e' \in \{\hat{f}_L^i(\hat{e}) | \hat{e} \in E_V^i\}$

where t^- and t are the time instances immediately before and after the arrival, departure or reconfiguration event. $C(e)$ is the number of times that the substrate link e appears in all selected substrate paths for the i'th VN. (f_N^i, f_L^i) and $(\hat{f}_N^i, \hat{f}_L^i)$ are the original and the reconfigured assignments for the i'th VN respectively.

B. Virtual network assignment problem description

Given a substrate topology G_s with the current stress level on each substrate link and node, upon the arrival of a VN request G_v^i, our goal is to assign substrate nodes and links to realize the requested virtual topology (as shown in Figure 2).

There are various ways to map a VN to the substrate network. The key issue is how to efficiently use substrate resources. Due to limitations on processing power, each substrate node could only be virtualized into a certain number of virtual nodes. Therefore, maintaining low stresses across all substrate nodes is important for the efficient use of network resources.

Maintaining low link stresses is also important in both provisioned substrate links (*i.e.,* substrate links with performance guarantees for its carried virtual links) and unprovisioned substrate links (*i.e.,* substrate links without performance guarantees). 1) For provisioned links, there is a provisioned capacity that could be allocated to virtual links. Reducing the stress will reduce the total bandwidth needed along that link. Furthermore, to provide guaranteed performance, special efforts are needed to isolate different virtual links within the same provisioned substrate link. Therefore, reducing the substrate link stress would reduce the above isolation efforts. 2) For unprovisioned links, reducing the stress will reduce the effects of cross traffic interferences since less virtual links are sharing the substrate link.

Furthermore, having a balanced stress across the substrate network is essential to avoid hot spots and reduce congestion. Therefore, the objective of the VN assignment problem is to maintain *low* and *balanced* stress among all substrate links and substrate nodes.

To evaluate the stress balancing performance, we define the node stress ratio (R_N) as the ratio of the maximum node stress and the average node stress across the whole substrate network. Similarly, the link stress ratio (R_L) is the ratio of the maximum link stress and the average link stress, *i.e.,*

$$R_N(t) = \frac{\max\limits_{v \in V_S} S_N(t,v)}{[\sum\limits_{v \in V_S} S_N(t,v)]/|V_S|} \quad (1)$$

$$R_L(t) = \frac{\max\limits_{e \in E_S} S_L(t,e)}{[\sum\limits_{e \in E_S} S_L(t,e)]/|E_S|} \quad (2)$$

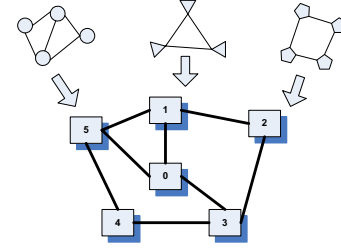Fig. 2.   VN assignment: Allocate substrate resources to VN nodes and links.

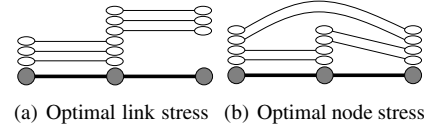(a) Optimal link stress   (b) Optimal node stress

Fig. 3.   Different virtual network assignments: (a) achieves the optimal link stress and (b) achieves the optimal node stress

The stress ratio is a proper metric to evaluate the load balancing performance since a smaller stress ratio indicates that the stress is more balanced. If the node (or link) stress ratio is 1, the network achieves optimal load balancing since all substrate nodes (or links) have the same stress.

Although the link stress ratio reflects the load balancing performance, it may not be sufficient to reveal the true efficiency of the VN assignment. Figure 3 illustrates the problem of the stress ratio. The substrate network has a 3-node 2-link topology (shown as shadowed circles and thick links). There are 6 single-link VNs (represented by ellipses and links connecting them) that are needed to be assigned to the substrate network. Both Figure 3(a) and 3(b) achieve perfect link stress balancing since their $R_L = 1$. However, the latter is not efficient in using the link resources, since its links have higher stress. This indicates that assignments with different efficiency could have the same R_L. The key factor here is the maximum link stress. Therefore, instead of using the stress ratio, we use the maximum link/node stress as the performance metrics, *i.e.,* we want to minimize both the maximum node stress and maximum link stress.

Due to the heterogeneity of the substrate topology, it may not be possible to achieve optimal stress for both substrate nodes and links. Again, using Figure 3 as an example, it is easy to verify that the assignment in Figure 3(a) minimizes the maximum link stress and its maximum link stress and maximum node stress are 3 and 6 respectively. Figure 3(b), on the other hand, minimizes the maximum node stress and the corresponding maximum link stress and maximum node stress are both 4. In fact, there is no such an assignment for these VNs that could achieve both the optimal node stress and optimal link stress. Therefore, instead of having a single term, the objective function of the VN assignment problem should be defined as the combination of the two:

$$\alpha \cdot \max\limits_{v \in V_S} S_N(a_i, v) + \beta \cdot \max\limits_{e \in E_S} S_L(a_i, e) \quad (3)$$

where a_i is the time instance immediately after the i'th VN arrival, $\alpha, \beta > 0$ are weights for the node stress objective and link stress objective respectively and are determined by the specific application scenario.

To summarize, the VN assignment problem for the i'th VN can be formalized as the following snapshot optimization problem upon the VN's arrival:

- Inputs:
  1) Substrate network information: The substrate topology G_S, snapshot of the current substrate node stresses $\{S_N(a_i^-, v) | \forall v \in V_S\}$ and link stresses $\{S_L(a_i^-, e) | \forall e \in E_S\}$, where a_i^- is the time instant immediately before the i'th VN arrival.
  2) The requested VN topology G_V^i
- Find the mappings from the the i'th VN topology to the substrate topology: f_N^i and f_L^i
- Minimizing

$$\alpha \cdot \max_{v \in V_S} S_N(a_i, v) + \beta \cdot \max_{e \in E_S} S_L(a_i, e)$$

In this paper, we consider two versions of the VN assignment problem: VN assignment without reconfiguration (VNA-I) and VN assignment with reconfiguration (VNA-II). The former refers to the case where the VN assignment is fixed for the whole VN lifetime and the latter refers to case where the VN assignment is allowed to changed during the VN lifetime. In the next two sections, we will present algorithms for both versions of the problem respectively.

IV. VNA-I: VN ASSIGNMENT WITHOUT RECONFIGURATION

For each VN request, if the mappings of the virtual nodes are given, then the VN assignment problem becomes an off-line load balancing routing problem where the source and destination are the ingress and egress nodes of each virtual link and each flow has a unit demand. This can be formulated as a well-known NP-hard unsplittable flow problem where the objective is to minimize the maximum link load [14]–[16]. The general VN assignment problem is even harder since the node mapping could be arbitrary and we want to optimize both node stress and link stress. Therefore, heuristic approaches are used to solve the problem. In this section, we will first describe a basic VN assignment scheme that acts as a building block for all other advanced algorithms. An improved VN assignment algorithm that subdivides the VN topology is then presented. Adaptive optimization strategies that achieve desirable tradeoffs between the node stress and link stress optimization objectives are also discussed.

A. A basic VN assignment algorithm

A naive way to perform VN assignment is to treat the node stress optimization and link stress optimization as two independent sub-problems and solve them sequentially. However, we will see from the simulation results presented later that this approach usually results in high link stress since the selected substrate nodes may be far away from each other. In this subsection, we propose a VN assignment algorithm
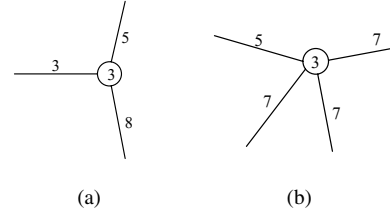

(a) (b)

Fig. 4. Neighborhood resource availability, numbers represent the stress

that considers both node and link stresses throughout the VN assignment process.

The key idea of our algorithm is to select a cluster of substrate nodes that are not only lightly loaded but also likely to result in low substrate link stresses when they are connected. This is achieved through a coarse-to-fine approach: First, a cluster center in the substrate network is identified based on the stress level in its neighborhood. Substrate nodes are then selected sequentially based on their distances from the previously assigned virtual nodes.

1) Cluster center localization: Since any directly connected virtual link of a virtual node will go through one of the directly connected substrate links of the corresponding substrate node, substrate nodes with high link stresses on all their directly connected links should be avoided. To quantify the stress level of a substrate node and its connected substrate links, we define the following neighborhood resource availability (NR):

$$\text{NR}(t, v) = [S_{\text{nmax}}(t) - S_N(t, v)] \cdot \sum_{e \in L(v)} [S_{\text{lmax}}(t) - S_L(t, e)] \quad (4)$$

where $S_{\text{nmax}}(t) = \max_{v \in V_S} S_N(t, v)$ and $S_{\text{lmax}}(t) = \max_{e \in E_S} S_L(t, e)$ are the maximum node stress and maximum link stress of the substrate network respectively. $L(v)$ is the set of substrate links directly connected to substrate node v.

The above definition of NR captures both the node stress and neighboring link stresses in the sense that a substrate node with a high NR means that both the node itself and some of its directly connected links are lightly loaded. The cluster center, which is the first selected substrate node, is therefore identified as the substrate node with the highest NR value. In the example shown in Figure 4, assume $S_{\text{nmax}}(t) = 5$ and $S_{\text{lmax}}(t) = 8$, then the NR for the substrate node in Figure 4(a) and 4(b) are 16 and 12 respectively. Therefore, the former substrate node is preferred even though it has a lower node degree.

2) Substrate node selection: After identifying the cluster center, the next step is to select the rest of the substrate nodes for the VN request. Rather than simply selecting nodes with minimum stresses, our node selection algorithm also considers the link stress by weighting the node stress based on its distances to other selected substrate nodes. We adopt the definition of distance from the shortest-distance path algorithm [17], which can effectively balance the network load among links and efficiently utilize network resources. Based on this notion, the distance of a single substrate path p at time t is defined as:

$$d(t,p) = \sum_{e \in p} \frac{1}{S_{\text{lmax}}(t) + \delta_L - S_L(t,e)} \tag{5}$$

where $\delta_L \ll S_{\text{lmax}}(t)$ is a small positive constant to avoid dividing by zero in computing the distance. And the distance between two substrate nodes u and v is defined as the minimum distance of all paths between them, *i.e.,*

$$D(t,(u,v)) = \min_{p \in P_S(u,v)} d(t,p) \tag{6}$$

Since the VN topology could be arbitrary, there could be a virtual link between any pair of virtual nodes. Therefore, to minimize the overall distance, the weight a virtual node is set to be the summation of distances to all previously assigned virtual nodes. Combining the distance and node stress, we use the following *node potential* as the criteria for substrate node selection:

$$\pi(t,v) = \frac{\sum\limits_{u \in V_A} D(t,(v,u))}{S_{\text{nmax}}(t) + \delta_N - S_N(t,v)} \tag{7}$$

where V_A is the set of selected substrate nodes for the same VN and $\delta_N \ll S_{\text{nmax}}(t)$ is a small positive constant to avoid dividing by zero. We choose $\delta_L = \delta_N = 1$ as the default value. The set of substrate nodes with the minimum potential are then selected.

After all substrate nodes are determined, we need to map the virtual nodes into the selected substrate nodes such that virtual nodes with higher degree are mapped to substrate nodes with higher NR. The intuition behind this is that virtual nodes with higher degree will setup more virtual links. All of these virtual links will go through some of the substrate link in the neighborhood of the corresponding substrate node. Therefore, the above matching tends to reduce the congestion and balance the link stress.

3) Substrate path selection: The last step of the basic VN assignment scheme is to connect the selected substrate nodes based on the virtual topology. This is a standard routing problem and we use the shortest-distance path algorithm [17] again to select the path with minimum distance (defined in Eq. (5)) between corresponding substrate nodes. Therefore, it is consistent with Eq. (6) in our node selection process. The detailed basic VN assignment algorithm is given in Algorithm 1. The basic VN assignment algorithm presented above can be used directly to allocate substrate resources for VN requests. However, since its node selection process does not take the VN topology into consideration, the basic algorithm may become inefficient when the VN has a sparse topology. Furthermore, the basic algorithm always compromises the node stress performance with the link stress performance regardless of the current network situation. More intelligent approaches are discussed next to address these issues.

B. Subdividing the virtual network

In the basic VN assignment algorithm, the requested VN is mapped to the substrate network as a whole unit. Small VNs

---

Algorithm 1 Basic VN assignment algorithm (Upon the i'th VN arrival)

INPUTS:
    $G_s = (V_S, E_S)$: substrate topology;
    $S(a_i^-, v), \forall v \in V_S$: current link stress;
    $S(a_i^-, e), \forall e \in E_S$: current node stress;
    $G_v^i = (V_V^i, E_V^i)$: VN topology;
OUTPUTS:
    $f_N^i(\hat{v}), \forall \hat{v} \in V_V^i$ and $f_L^i(\hat{e}), \forall \hat{e} \in E_V^i$

$V_A = \emptyset$ {Note: V_A is the set of selected substrate nodes}
$V_A = V_A \cup \{\arg\ \max_{v \in V_S} \text{NR}(a_i^-, v)\}$
for $i = 2, \ldots, |V_V^i|$ **do**
    $V_A = V_A \cup \{\arg\ \min_{v \in V_S - V_A} \pi(a_i^-, v)\}$
end for
Assign nodes in V_A to virtual nodes such that
$\text{NR}(f_n^i(\hat{u})) \geq \text{NR}(f_n^i(\hat{v}))$, *iff* $\text{Degree}(\hat{u}) \geq \text{Degree}(\hat{v})$
Find the shortest-distance path for all virtual links

---

have smaller footprints on the substrate network and are therefore easier to fit into isolated low stressed subregions in the substrate network. As an extreme case, if every VN contains only a single virtual link, then we have the most flexibility to allocate substrate network resources and can always achieve optimal link stress performance. As the VN topology becomes larger, it is more difficult to balance the load if we consider the whole VN at once. To utilize the flexibility of the small topology, we break up the VN into a number of connected sub-virtual networks (*subVN*). Each subVN is of a simple star topology and the connections between subVNs define the constraints in substrate node selection. Another advantage of subdividing is that by breaking down the big network into a number of small subVNs, computational requirements are reduced since less virtual nodes and links are considered in each subVN assignment process. Furthermore, since all subVNs have star topologies, the only possible links in the subVN are to/from the subVN center. Therefore, we can be more accurate in calculating the node potential in Eq. (7), since $D(t,(u,v))$ can be summed over exact virtual links instead of assuming virtual links to every previously assigned virtual nodes.

The subdividing algorithm works in a recursive way: starting from the original VN topology $G_V = (V_V, E_V)$, the algorithm finds the virtual node with the highest degree as the center of the subVN. All virtual nodes directly connected to the selected center and the corresponding virtual links form a subVN and are removed from the original VN. This process continues on the residual topology $G_R = (V_R, E_R)$ until all the virtual links have been assigned to a certain subVN. The detailed algorithm is given in Algorithm 2. Figure 5 shows an example of the subdividing process. For the original VN shown in Figure 5(a), the subdividing algorithm sequentially generates three subVNs as shown in Figure 5(b), 5(c) and 5(d) respectively.

The above process divides a VN topology into a number of subVNs with some inter-subVN constraints. Each subVN is then assigned to the substrate network based on the basic algorithm described in Section IV-A and the attached inter-

Algorithm 2 Subdividing algorithm

INPUTS: $G_V = (V_V, E_V)$
OUTPUTS: $G_{V(i)} = (V_{V(i)}, E_{V(i)}), i = 1, 2, \cdots$

$G_R = G_V, i = 0, G_{V(i)} = \emptyset$
while $G_R \neq \emptyset$ **do**
  $\hat{v} = \arg\ \max_{\hat{v}' \in V_R} \text{Degree}(\hat{v}')\}$
  $V_{V(i)} = V_{V(i)} \bigcup \{\hat{v}\}$
  for $\forall \hat{u} \in V_R$ **do**
    if $\overline{\hat{v}\hat{u}} \in E_R$ **then**
      $V_{V(i)} = V_{V(i)} \bigcup \{\hat{u}\}$, $E_{V(i)} = E_{V(i)} \bigcup \{\overline{\hat{v}\hat{u}}\}$
    end if
  end for
  $G_R = G_R - G_{V(i)}$, $i = i + 1$
end while

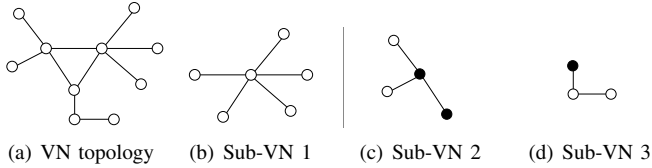(a) VN topology   (b) Sub-VN 1   (c) Sub-VN 2   (d) Sub-VN 3

Fig. 5. Subdividing the virtual network, constrained nodes are marked in black.

subVN constraints are handled as follows:

- Unconstrained subVN: In this case, there is no constraint on the subVN assignment so the subVN can be assigned to any location in the substrate network. Assume the requested VN topology is connected, then the only unconstrained subVN is the first one generated by the subdividing algorithm. The basic scheme can therefore be directly applied to this subVN. An unconstrained subVN is shown in Figure 5(b).

- Constrained subVN: In this case, the mappings of some virtual nodes in the subVN have already been determined by the previous assigned subVNs. Therefore, only nodes that are not constrained need to be assigned. Assume there are m unconstrained VN nodes, based on whether the subVN center is constrained or not, it can be further divided into the following two cases:

  - Center constrained: If the subVN center is constrained, then the node selection is based on the relative distance to the center. Specifically, the algorithm selects m substrate nodes with the least potential. A center constrained subVN is shown in Figure 5(c).
  - Center unconstrained: In this case, we need to determine the subVN center first by selecting the substrate node with the maximum NR. After that, we can use the same scheme as the center constrained case. A center unconstrained subVN is shown with Figure 5(d).

Note that since all subVNs have star topologies, the only possible virtual links are those connecting to the subVN center. Therefore, when calculating the potential in Eq. (7), V_A should be modified to contain only the subVN center.

C. Adaptive optimization strategies

We have shown in Section III-B that there are inherent tradeoffs between optimization objectives of node stress and link stress. To optimize the node stress, a simple way is to select substrate nodes with the least stress. However, this naive approach totally ignores the link stress in node selection and may result in poor link stress performance. This is because the selected nodes may be scattered far away from each other in the substrate network and choosing these nodes may result in excessively long substrate paths. The basic VN assignment scheme described in Section IV-A also aims at minimizing the maximum node stress and could achieve near optimal node stress and much better link stress performance, as will be shown later in the simulation results. Therefore, it can be viewed as a node optimization strategy (*node-opt*). On the other hand, if the objective is to optimize link stress only, then the node selection and link mapping should be solely based on the substrate link stress. Specifically, we use the same algorithm in Algorithm 1 but with a modified definition of the NR and the node potential as follows:

$$\hat{\text{NR}}(t, v) = \sum_{e \in L(v)} [S_{\text{lmax}}(t) - S(t, e)] \tag{8}$$

$$\hat{\pi}(t, v) = \sum_{u \in V_A} D(t, (v, u)) \tag{9}$$

This modified algorithm aims to optimize link stress performance and therefore is called the link optimization strategy (*link-opt*).

Based on the application context, either the node-opt or link-opt strategy can be used to perform VN assignment. As we discussed earlier, due to the heterogeneity of substrate topology, it may not be possible to achieve both the optimal node performance and optimal link performance. Therefore, instead of using either the node-opt or link-opt for all VN assignments, we use an adaptive strategy that chooses between the node-opt and link-opt for any given VN request adaptively.

The node stress ratio and link stress ratio defined in Eq. (1) and (2) provide a straightforward way to identify the relative performance of link and node stress balancing. Specifically, if $R_L(t) > R_N(t)$, the substrate link stress is more unbalanced. Otherwise, the substrate node stress is more unbalanced. Upon the arrival of a new VN request, the adaptive scheme determines if the node stress or link stress is more unbalanced and performs the node-opt or link-opt strategy accordingly. More generally, we can set an *adaptive threshold* $\eta \in [0, +\infty)$, and use the adaptive scheme shown in Algorithm 3. The adaptive strategy is especially effective if the objective of the VN assignment is in the form of Eq. (3). In that case, the value of η could be determined based on values of α and β. For example, we can set $\eta = \alpha/\beta$.

V. VNA-II: VN ASSIGNMENT WITH RECONFIGURATION

In the dynamic process of VN assignment, network conditions change over time due to the arrival and departure of VNs. As a result, VN assignment without reconfiguration may
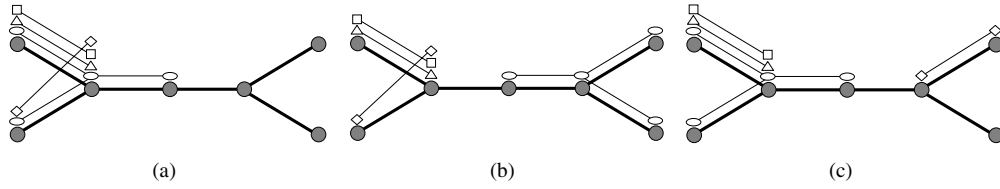
Fig. 6. Effects of different reconfiguration orders

Algorithm 3 Adaptive optimization strategy (Upon the i'th VN arrival)

---

if $R_L(a_i^-) > \eta \cdot R_N(a_i^-)$ **then**
    Perform link-opt
else
    Perform node-opt
end if

---

lead to inefficient resource utilization where some part of the substrate network can become excessively loaded while others are under-utilized. Similar to the case of rerouting [18], reconfiguring existing VNs can help to improve performance under dynamic situations. However, periodically reconfiguring all existing VNs is not desirable for a number of reasons.

First, VN reconfiguration is much more expensive than rerouting. The cost of reconfiguration includes both the computational cost and the service disruption cost. Computational cost refers to the expenses involved in recomputing the VN assignment and therefore is proportional to the frequency of VN reconfiguration. Service disruption cost is incurred because the normal operation of a VN is affected when it is switched from one assignment to another. Unlike flow rerouting where at most one path change happens per rerouting event, the reconfiguration of a single VN may result in more substantial changes involving both *node switching* and multiple *path switching*. Node switching happens when the assignment of a virtual node is changed from one substrate node to another. Path switching happens when the assignment of a virtual link is changed from one substrate path to another. The disruption of the reconfiguration is therefore significant if all VNs are reconfigured periodically. Based on the above discussion, we quantify the reconfiguration cost as follows:

$$C = w_1 \cdot R_{\text{recfg}} + w_2 \cdot R_{\text{node}} + w_3 \cdot R_{\text{path}} \quad (10)$$

where R_{recfg} is the reconfiguration rate, defined as the number of reconfiguration events per time unit. R_{node} and R_{path} are the node and path switching rates, defined as the number of substrate node and path switches per time unit respectively. $w_1, w_2, w_3 > 0$ are weights of each cost component and are determined by the specific application scenario.

Furthermore, the reconfiguration order of different VNs will affect the VN assignment performance. If there is a limit on the number of reconfigurations, reconfiguring VNs that are on maximum stressed links/nodes will be more likely to reduce the maximum stress than reconfiguring other VNs. Therefore, VNs that are on the maximum stressed links/nodes are *critical*

to the performance. Even if there is no limit on the number of reconfigurations, reconfiguring non-critical VNs before critical ones may take up resources that could otherwise be used by critical VNs to improve the performance.

Figure 6 illustrates the effects of reconfiguring different VNs. Assume that the network is originally in a state shown in Figure 6(a), where there are 4 VNs on the substrate network (three 2-node VNs and one 3-node VN). The 2-node diamond VN is non-critical since it is not carried by any maximum stressed link. In contrast, the other two 2-node VNs and the 3-node VN are all critical since they use the maximum stressed link. If the 3-node VN reconfigures first, then the network enters the state shown in Figure 6(b). The maximum link stress is reduced from 3 to 2. On the other hand, if the 2-node diamond VN reconfigures first, then the network state is shown in Figure 6(c) and the maximum link stress is unchanged. Obviously, the former reconfiguration is more desirable.

Based on this notion of criticality, we developed a selective reconfiguration scheme to perform efficient reconfiguration by giving higher priority to critical VNs. The key idea is to reconfigure only a subset of the existing VNs that are most critical in reducing the maximum stress instead of reconfiguring all existing VNs. The selective reconfiguration algorithm has two components: a global marking process and per VN reconfiguration processes.

1) Global marking: The marking process periodically (with the marking period τ_M) compares the node stress ratio R_N and link stress ratio R_L (Eq. (1) and (2)) to determine which one is more unbalanced. The algorithm then identifies a set of critical substrate nodes (or links) accordingly as follows:

$$V_C(t_M) = \{v | S(t_M, v) \geq (1 - \theta) \cdot S_{\text{nmax}}(t_M), v \in V_S\}$$
$$E_C(t_M) = \{e | S(t_M, e) \geq (1 - \theta) \cdot S_{\text{lmax}}(t_M), e \in E_S\}$$

where t_M is the time instance of the marking event. $\theta \in [0, 1]$ is the *reconfiguration threshold* that controls the stress level to be considered as critical. For example, $\theta = 0$ means that only the maximum stressed nodes/links are considered critical and $\theta = 1$ means that all links/nodes are considered critical. Next, all VNs that are currently using the critical nodes (or links) are marked for reconfiguration until the next marking event. The detailed marking algorithm is given in Algorithm 4.

2) Per VN reconfiguration: The above global marking process only determines which VNs are allowed to reconfigure. The actual reconfiguration process happens at each individual VN. Upon the VN arrival, the VN assignment algorithm presented in Section IV is performed to allocate substrate

Algorithm 4 Global marking algorithm

---

if $R_L(t_M) > R_N(t_M)$ **then**
    Identify critical substrate links $E_C(t_M)$
    Mark all VNs on critical links: $\{i|\exists e \in E_V^i, f_L^i(e) \bigcap E_C(t_M) \neq \emptyset\}$
else
    Identify critical substrate nodes $V_c(t_M)$
    Mark all VNs on critical nodes: $\{i|\exists v \in V_V^i, f_N^i(v) \in V_C(t_M)\}$
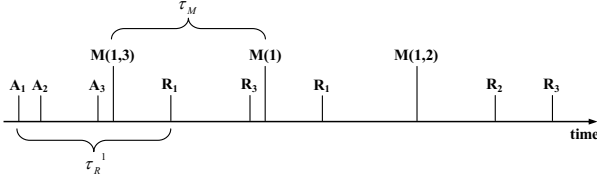**end if**

Fig. 7. Global marking and per VN reconfiguration events with related time scales. A, M and R are VN arrival, marking and reconfiguration events respectively. Numbers besides A, R and τ_R index the VN. Numbers besides M show the indexes of marked VNs. τ_M and τ_R are the marking period and reconfiguration period respectively.

resources. Furthermore, each existing VN also periodically checks (with *reconfiguration period* τ_R^i for the i'th VN) if it is selected by the marking process. If it is the case, then the VN reapplies the VN assignment algorithm to compute the new assignment. Otherwise, the VN keeps its current assignment. The previous marking and reconfiguration events and the corresponding time scales are illustrated in Figure 7.

The above algorithm reduces the reconfiguration cost by selecting only a subset of VNs for reconfiguration. By allowing only critical VNs to reconfigure, it also optimizes the reconfiguration order since non-critical VNs are less likely to be reconfigured. Furthermore, since the marking process happens periodically, the algorithm can adapt to dynamic situations by selecting different sets of critical VNs accordingly.

VI. PERFORMANCE EVALUATION

In this section, we first evaluate the performance of VN assignment algorithms without reconfiguration. Next, we investigate the performance of our selective VN reconfiguration algorithm under different parameter settings. Finally, we examine the VN assignment performance over a wide range of network settings.

A. Simulation settings and performance metrics

We have implemented a discrete event simulator for VN assignment. The substrate network is a 100-node 316-link random topology generated by the GT-ITM tool [19]. We assume that VN requests arrive in a Poisson process with an average rate λ VNs per time unit, and the lifetime of VNs follows an exponential distribution with an average of $\mu = 275$ time units. Unless otherwise specified, the VN topologies are random and the VN size is uniformly distributed from 2 to 10. The average VN connectivity (*i.e.,* probability that there is a virtual link between a pair of virtual nodes) is fixed to be 50%. To evaluate the performance of a VN assignment algorithm,

we use Poisson sampling, with an average inter-arrival time of 1 time unit, to estimate the time-averaged maximum node stress (N_{max}) and maximum link stress (L_{max}) as follows:

$$N_{max} = \frac{\sum_i \max_{v \in V_S} S(t^{(i)}, v)}{M}$$

$$L_{max} = \frac{\sum_i \max_{e \in E_S} S(t^{(i)}, e)}{M}$$

where $t^{(i)}$ is the time instance of the i'th sample and M is the total number of samples.

Each data point in the simulation results is obtained by running the simulation until 80,000 VN requests have been serviced. Furthermore, to reduce the effects of transient simulation results, we start to collect data after the first 40,000 VN requests. The results in this section are shown with their 95% confidence intervals. Note that the confidence intervals are very small, sometimes hardly visible on the plots due to the relatively large number of samples.

B. VN assignment performance without reconfiguration

1) Effects of the offered load: In the first experiment, we increase the VN arrival rate λ while fixing the average VN lifetime and show the performance of the following static VN assignment algorithms:

- *Basic*: The basic VN assignment algorithm presented in Section IV-A
- *SubVN*: The basic VN assignment with the enhancement of subdividing the VN topology
- *Adaptive*: The SubVN based scheme with the adaptive optimization strategy where the adaptive threshold $\eta = 1$

For comparison purposes, we also show results of the least loaded algorithm (*least-load*) that treats link stress optimization and node stress optimization separately by selecting the least loaded substrate nodes as the virtual nodes and connecting them using the shortest distance path algorithm.

Since there is no blocking in the system (*i.e.,* VN requests are never rejected for service), both L_{max} and N_{max} grow linearly with λ (as shown in Figure 8). Note that under a fixed λ, the arrival rate of VN nodes is fixed. Therefore, the average substrate node stress is the same for any VN assignment algorithm. Figure 8(a) shows that the least-load, basic and subVN based schemes can all achieve similar N_{max} which is very close to the average node stress (shown in dashed line in Figure 8(a)). Since N_{max} could not be lower than the average node stress, this result indicates that all of the above three algorithms achieve near optimal node stress performance. The adaptive scheme has higher N_{max} since it trades off node stress performance for link stress performance.

The link stress performance in Figure 8(b) shows that the least-load scheme performs poorly in terms of L_{max}. The basic VN assignment scheme effectively reduces L_{max} by more than 30%. The subVN based scheme further reduces L_{max} by subdividing the VN topology. The adaptive scheme has the
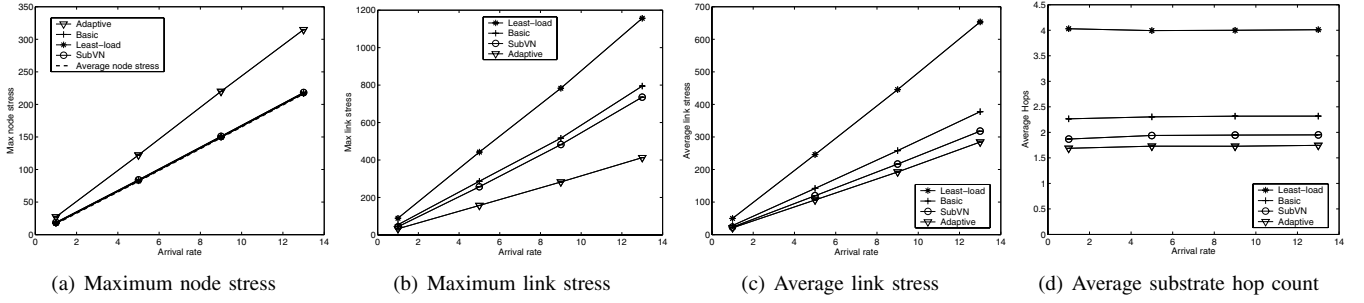
(a) Maximum node stress   (b) Maximum link stress   (c) Average link stress   (d) Average substrate hop count

Fig. 8.   VN assignment performance under increasing offered load

best link stress performance and its L_{max} is about 65% lower than the least-load scheme.

Results of the average link stress and average substrate hop count for virtual links are given in Figure 8(c) and Figure 8(d) respectively. They clearly show that our VN assignment algorithms reduce the resources usage by using shorter substrate paths. The least-load scheme, in contrast, does not consider the link stress in node selection so the selected substrate nodes could be far away from each other. Therefore, the least-load algorithm has much higher average link stress and average substrate hop count. We also observe in Figure 8(d) that the average substrate hop count is fixed under different VN arrival rate. The reason is that all of our VN assignment schemes are based on the relative orders of node/link stress, NR, or node potential, and these relative orders do not change with the offered load.

From this experiment, we can see that the basic VN assignment algorithm is consistently better than the least-load algorithm such that it can achieve near optimal node stress performance with much lower and more balanced link stresses. The adaptive VN assignment algorithm effectively balances the link/node stress objectives and significantly reduces the maximum link stress from the basic VN assignment algorithm. Furthermore, the virtual network built by our VN assignment algorithms has better quality than the least-load algorithm since the average virtual link length in substrate hop counts is significantly lower.

2) Benefits of the subVN scheme: The next experiment focuses on the subVN based scheme. Figure 9 compares L_{max} of both the basic scheme and the subVN scheme when the average VN connectivity is increased from 0.2 to 1.0 and the VN arrival rate is fixed at 10 per time unit. Both curves appear to be linear to the average VN connectivity since the latter linearly determines the arrival rate of virtual links. The subVN scheme achieves consistently lower L_{max} than the basic scheme but the differences between these two reduces from about 33% to almost zero as the VN connectivity increases from 0.2 to 1. Recall that the key improvement of the subVN scheme over the basic scheme is to perform node selection based on actual VN topology instead of assuming full VN connectivity. Obviously, the above difference becomes less significant as the actual VN connectivity increases. This experiment indicates that the subVN algorithm is more effective
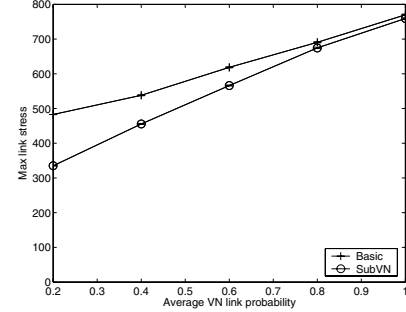

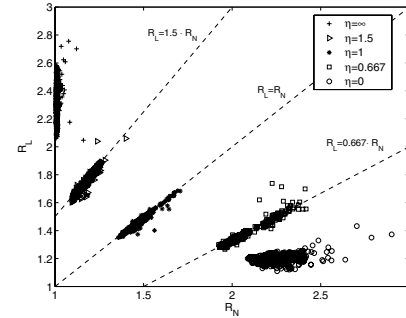Fig. 9.   Effects of VN connectivity

Fig. 10.   Tradeoffs between link stress and node stress performances

for VNs with sparse connectivity. For the densely connected VNs, the basic scheme could be used instead to simplify the VN assignment process without losing much performance. For the rest of the paper, we use the subVN scheme by default.

3) Effectiveness of adaptive optimization strategies: To demonstrate the effectiveness of adaptive optimization strategies in VN assignment, we use an R_N vs. R_L graph: During each sampling instance in the simulation, we plot a point in the R_N vs. R_L graph using the corresponding values of (R_N, R_L) as the coordinates. Therefore the location of the (R_N, R_L) point indicates its relative performance of node stress balancing and link stress balancing. Specifically, all points located above the line of $R_L = R_N$ achieve better node stress balancing than link stress balancing, while points located below the line of $R_L = R_N$ have better link stress balancing performance. In this experiment, we fix the VN arrival rate at 15 per time unit and vary the adaptive threshold
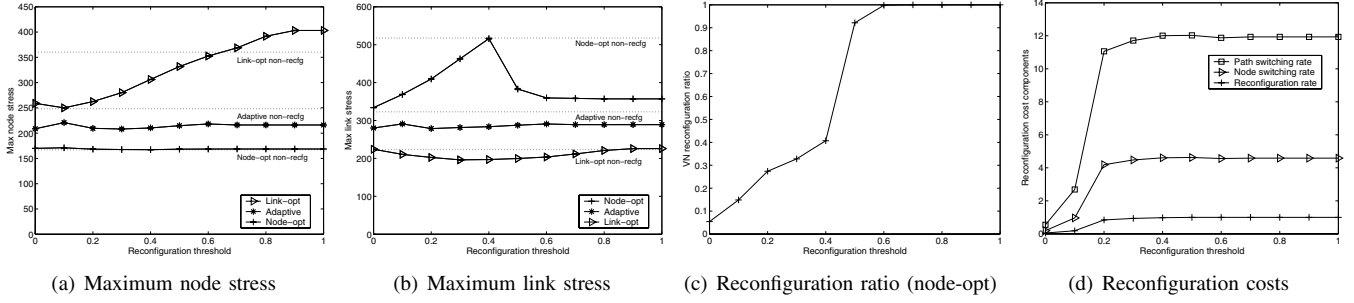
| (a) Maximum node stress | (b) Maximum link stress | (c) Reconfiguration ratio (node-opt) | (d) Reconfiguration costs |

Fig. 11. Performance vs. reconfiguration threshold

η to be $0, 0.667, 1, 1.5, \infty$ respectively and plot (R_N, R_L) points obtained from all sampling instances in Figure 10. Note that $\eta = 0$ and $\eta = \infty$ represent the pure link optimization (link-opt) and pure node optimization (node-opt) strategy respectively.

Figure 10 shows that VN assignment algorithms with different values of η form distinctive clusters in the R_N vs. R_L graph. Points of the node-opt strategy ($\eta = \infty$) are all located near the line of $R_N = 1$, indicating that the node-opt strategy achieves the optimal node stress balancing. However, it has very unbalanced link stress than all other cases since its points have the highest R_L values. In contrast, the link-opt strategy ($\eta = 0$) has the lowest R_L values and therefore has the best link stress balancing performance. All intermediate values of adaptive threshold η result in different tradeoffs between node and link stress performance: As η increases from 0 to ∞, the VN assignment algorithm puts more emphasis on node stress optimization and less on link stress optimization.

Furthermore, points of $\eta = 0.667, 1, 1.5$ are closely located along their corresponding reference lines of $R_L = 0.667R_N$, $R_L = R_N$ and $R_L = 1.5R_N$ respectively, except for a small number of outlying points that occur during the transients. This indicates that the adaptive threshold η is able to effectively control the exact tradeoff between node and link stress balancing. For the rest of this section, we will only show the results of three representative optimization strategies: link-opt ($\eta = 0$), node-opt($\eta = \infty$) and adaptive ($\eta = 1$).

C. VN assignment performance with reconfiguration

Experiments in this subsection examine the performance of the selective VN reconfiguration algorithm. We choose the same value for both the marking period τ_M and per VN reconfiguration period τ_R^i so that critical VNs could have a chance to reconfigure promptly. Note that although their periods are the same, the marking and reconfiguring events are not synchronized since the latter is determined by the arrival time of each individual VN.

1) Effects of reconfiguration threshold: We first examine the effects of the reconfiguration threshold θ. Figure 11(a) and 11(b) show the performance the VN reconfiguration with node-opt, link-opt and adaptive strategies as θ is increased from 0 to 1. Note that $\theta = 0$ means only VNs using the highest stressed substrate nodes/links are allowed to reconfigure while

$\theta = 1$ means all VNs are allowed to reconfigure. Results for the corresponding non-reconfiguration schemes are also shown in dotted lines.

Our first observation is that VN reconfiguration generally achieves better L_{max} and N_{max} performance than the corresponding non-reconfiguration schemes. Figure 11(a) shows that N_{max} of the link-opt reconfiguration scheme is significantly lower than its non-reconfiguration counterpart when θ is small. When θ increases from 0 to 0.1, N_{max} of the link-opt scheme decreases slightly since more critical VNs are allowed to reconfigure. When θ is further increased, more non-critical VNs are reconfigured and take up substrate resources that could otherwise be used by critical VNs to reduce N_{max}. Therefore, N_{max} eventually increases as θ increase. The adaptive strategy demonstrates similar patterns but with a much smaller fluctuation magnitude since its node stresses are more balanced. Reconfiguration of the node-opt scheme has little effects on N_{max} performance since the node-opt scheme always achieves near optimal node stress performance.

In terms of the L_{max}, Figure 11(b) shows a more complicated picture for the node-opt scheme. Its L_{max} increases with θ when $\theta \leq 0.4$ since more non-critical VNs are competing with critical VNs. However, L_{max} has a sudden drop as θ increase from 0.4 to 0.5. To understand this phenomena, Figure 11(c) gives the average ratio of VNs that perform the reconfiguration in the node-opt strategy. It shows that the percentage increases from 45% to more than 90% when θ increase from 0.4 to 0.5. This indicates that there are about 50% of the VNs that are only located on most lightly loaded substrate links and reconfiguring them would make more room to accommodate critical VNs to reduce L_{max}. Therefore, when $\theta \geq 0.5$, almost all VNs are allowed to reconfigure and L_{max} is reduced.

Figure 11(d) shows three reconfiguration cost components (defined in Eq (10)) and all of them increase with the reconfiguration threshold θ. The changes are especially dramatic for small θ values. Therefore, a small θ should be used to keep reconfiguration cost low. Fortunately, as we have seen in both Figure 11(a) and 11(b), having a small θ could achieve most of the benefits of dynamic reconfiguration as long as we only allow critical VNs to reconfigure. The appropriate value of θ should be determined by the substrate topology and the resulting stress distribution. We set $\theta = 0$ in the rest

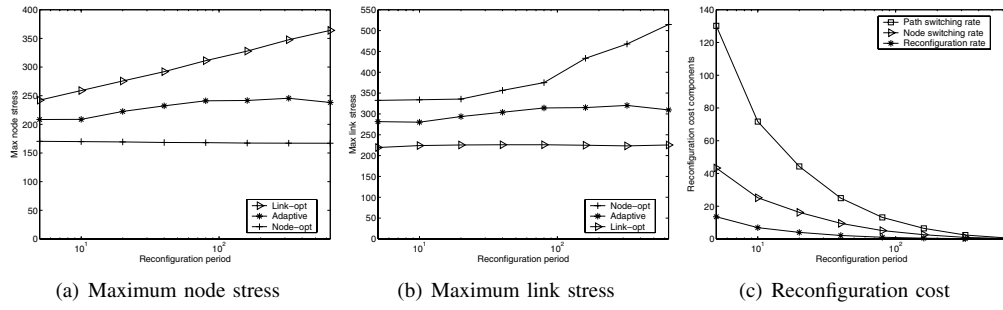(a) Maximum node stress (b) Maximum link stress (c) Reconfiguration cost

Fig. 12.   Performance vs. reconfiguration period

of the paper since it gives the best overall performance for our simulation topology.

2) Effects of reconfiguration period: Another important parameter of the VN reconfiguration algorithm is the reconfiguration period τ_R, which determines the frequency of reconfiguration. Figure 12 gives the performance of the reconfiguration algorithm when τ_R is increased from 5 to 640 time units. Figure 12(a) shows that N_{max} of both the link-opt and adaptive strategies increases with τ_R. The node-opt strategy is not sensitive to τ_R since it can always achieve near optimal node stress performance. Figure 12(b) shows that L_{max} of both the node-opt and adaptive strategies increases with τ_R. Therefore, having a small τ_R will achieve better overall performances. In terms of reconfiguration cost, however, having very small reconfiguration period is not desirable since it would result in excessively high costs as indicated by Figure 12(c), which shows the reconfiguration cost components for the adaptive strategy. The cost curves of the other two optimization strategies are similar. Considering the performance and cost together, $\tau_R = 10$ is a good choice since it can achieve most of the benefits of reconfiguration without excessively high cost.

D. Effects of different network settings

The final set of the experiments examines the performance of VN assignment algorithms under a wide range of network settings. Due to space limitation, we only show results for VN assignment without reconfiguration.

1) Effects of substrate network connectivity: Figure 13 shows the VN assignment performance when the average substrate node degree is increased from 3 to 19. We observe that both N_{max} and L_{max} decrease as the substrate network connectivity increases and the changes in L_{max} are more dramatic. Furthermore, the absolute performance differences among different VN assignment schemes are reduced as the substrate network becomes more densely connected.

In terms of N_{max}, increasing the substrate connectivity means more substrate nodes become close to each other and, therefore, the node selection scheme will not concentrate only on a small number of substrate nodes. As a result, the node stress becomes more balanced. In terms of L_{max}, when the substrate network is more connected, shorter paths can be found between substrate nodes and the effects of node selection on link stress balancing become less significant.
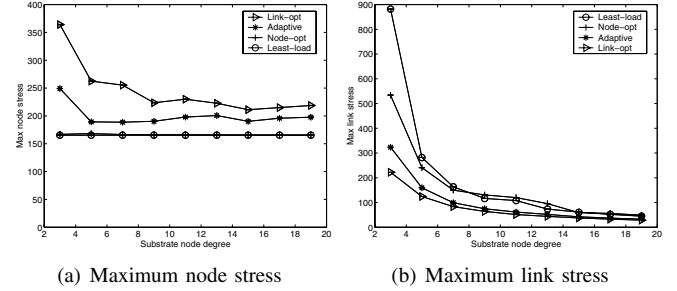

(a) Maximum node stress   (b) Maximum link stress

Fig. 13.   Performance vs. substrate connectivity

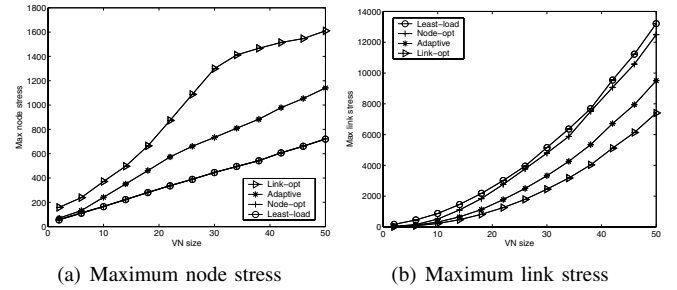(a) Maximum node stress   (b) Maximum link stress

Fig. 14.   Performance vs. VN size

Therefore, when the substrate network is densely connected, even the naive least-load algorithm achieves acceptable link stress performance. As a summary, this experiment indicates that the proposed VN assignment scheme is more effective when the substrate network has sparse connectivity.

2) Effects of VN size: This experiment studies the performance of VN assignment algorithms under different VN sizes. Figure 14 shows the results when the maximum VN size increases from 2 to 50. Note that the substrate network is fixed at 100 nodes, increasing VN sizes makes the node stress balancing easier since larger VNs are less sensitive to their node selection. As an extreme example, if the VN size is the same as the total number of substrate nodes, then all algorithms achieve perfect node stress balancing. This explains the reduced N_{max} increasing rate for both the link-opt and adaptive strategies at large VN sizes in Figure 14(a).

In terms of L_{max}, both the adaptive and link-opt strategies have significantly better performance than the least-load scheme. The node-opt strategy also performs better than the
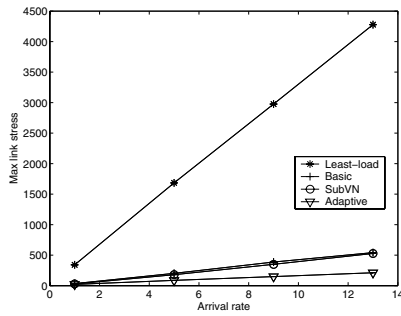
Fig. 15.   Maximum link stress (transit-stub substrate topology)

least-load scheme, but the differences are small especially for larger VNs. Again, this is because for larger VNs, the effects of node selection are less important, and therefore the node-opt strategy is more similar to the least-load scheme.

3) Multi-domain substrate network topology: To see the performance of the VN assignment algorithms under multi-domain substrate topology, we used a 124-node 510-link transit-stub substrate topology generated by the GT-ITM tool instead of the random topology. The substrate topology has a single 4-node transit domain and each transit node is attached by three 10-node stub domains. Figure 15 shows that L_{max} of the least-load scheme in this scenario is almost 8 times higher than the basic VN assignment algorithm. The differences are significant as compared to the corresponding results on random substrate topology (Figure 8(b)). This is because the substrate nodes with the lowest stress may be scattered in different domains and the least-load algorithm would select them for a single VN request. As a result, the transit-stub links become highly loaded because they are the only links that connect different domains. The proposed VN assignment scheme, in contrast, avoids hot spots by allocating resources for a single VN within the same domain.

VII. Concluding Remarks

Network virtualization is a promising way to de-ossify the current Internet by providing a shared platform for a variety of new network services and architectures. A major challenge in building the diversified Internet is to perform efficient and on-demand VN assignment.

In this paper, we developed a basic scheme for VN assignment without reconfiguration and use it as a building block for all other advanced algorithms. Subdividing heuristics and adaptive optimization strategies are presented to further improve the performance. We also developed a selective VN reconfiguration scheme that prioritizes the reconfiguration for the most critical VNs.

We evaluate the performance of the proposed VN assignment algorithms through extensive experiments and our findings are summarized as follows:

- The basic VN assignment algorithm performs consistently better than the least-load algorithm.
- The benefits of subdividing the VN topology are more significant when the VN topology is sparse.

- The advantage of proposed VN assignment algorithms is more prominent when the substrate network is sparsely connected.
- The proposed VN assignment algorithms can effectively avoid hot spots or congestion in the substrate network (such as the transit-stub links).
- For VN reconfiguration, reconfiguring only a subset of the existing VNs that are most critical achieves most of the benefits of dynamic reconfiguration while keeping a low cost. Small reconfiguration threshold and reconfiguration period should be used but their exact values should be determined by the specific substrate topology to achieve the desired tradeoff between performance and cost.

References

[1] L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[2] D. Taylor and J. Turner, "Towards a diversified internet," November 2004.

[3] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: a network design problem for multicommodity flow," in *Proc. ACM symposium on Theory of computing (STOC)*, 2001, pp. 389–398.

[4] A. Haque and P.-H. Ho, "Design of survivable optical virtual private networks (O-VPNs)," in *Proc. the 1st IEEE International Workshop on Provisioning and Transport for Hybrid Networks*, 2004.

[5] W. Szeto, Y. Iraqi, and R. Boutaba, "A multi-commodity flow based approach to virtual network resource allocation," in *Proc. GLOBECOM: IEEE Global Telecommunications Conference*, 2003.

[6] Q. Fang, J. Cobb, and E. Leiss, "A pre-selection routing scheme for virtual circuit networks," in *Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, 2000.

[7] Y. Azar, A. Z. Broder, and A. R. Karlin, "On-line load balancing," in *Proc. IEEE Symposium Foundations of Computer Science*, 1992, pp. 218–225.

[8] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, "On-line load balancing with applications to machine scheduling and virtual circuit routing," in *Proc. ACM symposium on Theory of computing*, 1993, pp. 623–631.

[9] Y. Azar, B. Kalyanasundaram, S. A. Plotkin, K. Pruhs, and O. Waarts, "Online load balancing of temporary tasks," in *Proc. Workshop on Algorithms and Data Structures*, 1993, pp. 119–130.

[10] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed.   Prentice-Hall, 1992.

[11] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection algorithms for replicated web servers," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 3, pp. 44–50, 1998.

[12] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[13] S. Shi and J. S. Turner, "Placing servers in overlay networks," in *Proc. Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, 2002.

[14] P. Raghavan and C. D. Thompson, "Provably good routing in graphs: regular arrays," in *Proc. ACM symposium on Theory of computing*, 1985, pp. 79–87.

[15] J. M. Kleinberg, "Approximation algorithms for disjoint paths problems," Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[16] S. G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," in *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 1997, p. 426.

[17] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proc. IEEE ICNP*, 1997, pp. 191–202.

[18] E. Wong, A. K. M. Chan, and T.-S. Yum, "A taxonomy of rerouting in circuit-switched networks," *IEEE Communications Magazine*, vol. 37, no. 11, pp. 568–582, 1999.

[19] E. W. Zegura, K. Calvert, and B. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE Infocom*, 1996.