Virtual Network Scheduling Design

H. Bai¹, F. Gu², J. Crichignoi³, S. Khan⁴, N. Ghani¹

¹Univ. of South Florida, ²Univ. of New Mexico, ³Northern New Mexico College, ⁴North Dakota State University

Abstract—Network virtualization provides an effective means of supporting multiple client infrastructures over a physical substrate. Now over the years, many different provisioning algorithms have been developed for network virtualization services, most recently driven by the growth in new cloud-computing paradigms. However, as user demands continue to evolve, there is growing need to schedule virtual network requests at future time instants. This is a very complex problem given the high dimensionalities involved. Along these lines this paper solves one of the first optimization formulations for virtual network scheduling and compares the findings to some advanced heuristics.

Index Terms—Virtual network, scheduling, advance reservation, cloud scheduling

I. INTRODUCTION

Network virtualization services are used to host multiple abstract client infrastructures over shared physical substrates. Overall, these offerings span a very wide range and include both virtual private network (VPN) and virtual network (VN) designs. In particular, VPN services map/embed connection links between fixed network locations (client sites) to build separate overlays [1]. By contrast, VN services are more flexible and allow client sites to be mapped to different network nodes, i.e., added node placement dimension [2]. Moreover, these latter types have been generalized to include data-center (computing, storage) resources and hence are well-suited for improving the scalability and reliability of emerging cloud-based paradigms [3].

Over the years, researchers have studied a wide range of VPN and VN provisioning schemes, including optimization and heuristic-based strategies, see surveys in [2],[3]. Nevertheless, most of these contributions assume on-line or batch requests arrivals. For example, [4] collects incoming VN requests in a given interval and optimizes them in a batch manner. However, as cloud services continue to evolve, there is an growing need to provision network virtualization services at future time instants, i.e., scheduling. For example, such capabilities can help pre-reserve distribution and storage resources to support large events, e.g., FIFA World Cup, NFL Superbowl, televised addresses, product launches, etc.

Now point-to-point connection scheduling has been well-studied over the years [5]. Further efforts have also looked at scheduling more complex multi-cast connections [6]. However, very few studies have looked at scheduling design for larger VPN or VN demands. For example, [7] presents one of the first studies on VN scheduling, termed as the virtual overlay network scheduling (VONS) problem. Namely, an integer linear programming (ILP) model is developed (but not solved) along with some basic heuristic strategies. Meanwhile the VN scheduling problem has been even less focus. This topic is particularly challenging given the added dimensionality of VN node placement in the time domain.

In light of the above, this paper presents an indepth study of scheduling design for network virtualization. In particular, the VPN scheduling problem is treated here, assuming fixed client (overlay) node locations, as per [7]. A revised ILP model is first presented and solved to provide a baseline for gauging performance. Improved heuristic strategies are then proposed using re-routing methods to improve resource utilization. In particular, re-routing is a very viable option in scheduling scenarios as future requests are inactive and hence can be re-assigned without disrupting active users.

Overall, this paper is organized as follows. Section II first presents a brief overview of the existing work on network virtualization provisioning and network scheduling. Section III then presents the proposed optimization formulation and the associated re-routing heuristics are detailed in Section IV. Finally, detailed performance evaluation results are presented in Section V using network simulation and, overall conclusions and future directions are presented in Section VI.

II. BACKGROUND

VN embedding is an NP-hard problem that involves mapping VN nodes to substrate nodes and VN links to substrate connections [3]. Owing to this complexity, researchers have proposed a wide range of solutions here. In particular, several mixed integer linear programming (MILP) formulations have been tabled to minimize resource utilization across VN requests or maximize revenues [2],[4]. However, since these models are only solvable for very small networks, heuristic strategies have also been developed using single and two-stage mappings. The former algorithms first embed all the VN nodes before routing VN link connections [8], whereas the latter jointly map VN nodes and links [4],[9]. In general, single-stage mappings give improved resource efficiency and higher revenues.

Meanwhile, the area of connection scheduling, or advance reservation (AR), has also been well-studied over the years, see detailed survey in [5]. For example, researchers have defined many different AR service models using combinations of fixed/variable connection start/stop times and/or durations [10]. A wide spectrum of related AR scheduling algorithms have also been developed to provision these service types, using both optimization and heuristic strategies. In particular, the former techniques assume time-slotted arrivals/departures and have considered various optimization objectives, e.g., minimizing resource utilization, maximizing the number of accepted requests, etc [11],[12].
Furthermore, some efforts have also looked at re-routing of scheduled demands to accommodate more requests [13],[14],[15]. The aim here is to improve resource utilization by re-routing future inactive connections, i.e., non-disruptive. Most of these strategies employ graph-based heuristic methods and try to achieve a tradeoff between the number of re-routing attempts and blocking reduction. Overall, findings show that these schemes can give notable improvements versus regular (non-re-routing) approaches. Now [15] also considers ILP re-optimization (after each request) to provide a hypothetical bound on re-routing performance. However, since the timeline dimension induces very high variable counts, global optimization of non-active requests quickly becomes intractable for even modest network sizes and/or lookahead timeslots. Hence a “dynamic” ILP model is also proposed to optimize over a reduced subset of inactive overlapping reservations for a new request. Overall, findings show notable blocking reduction versus heuristic strategies here.

Finally, researchers have also investigated scheduling of non-point-to-point demands. For example, [6] considers multicast scheduling design, whereas [7] introduces the VONS problem for scheduling more generalized VPN requests with arbitrary user-specific topologies. An associated ILP model is also presented in [7] to minimize resource consumption for a-priori demands (but not solved due to extremely high variable counts arising from the timeline dimension). Instead, several graph-based heuristics are presented to achieve resource minimization and load-balancing, and findings show lower blocking rates with the latter (albeit with higher utilization).

Overall, the scheduling of network virtualization services is a relatively new and unexplored area. Moreover, the need for related solutions here is expected to grow as providers look for improved methods to support projected/scheduled client demands. Hence it is important to develop realistic optimization models to bound achievable performance in such settings. Improved heuristic strategies are also critical to deliver more scalable solutions for larger real-world networks. These issues are now addressed further.

### III. OPTIMIZATION FORMULATION

A new ILP model is now presented for scheduling VPN requests. The VPN demand model here is essentially the same that in [7], i.e., where client end-point nodes are fixed to specific network nodes. (Note that the more challenging case of VN scheduling with variable node placement is left for future study). Now [7] introduces the VONS problem and presents a global ILP formulation to minimize its resource consumption for a predefined a-priori set of requests. Nevertheless, owing to excessive ILP variable counts, this formulation is not solved. As a result there is still a need to develop an ILP solution to provide some realistic bounds on scheduling performance for VPN demands, even if for relatively small networks.

In light of the above, a simplified dynamic ILP setup is proposed here for VPN scheduling. Namely, “dynamic” implies that arriving requests are optimized in a sequential on-time manner on per request basis by running an ILP over the residual capacity graph, i.e., versus optimizing complete batch a-priori demand sets. Note this is similar to the dynamic (connection-level) ILP scheme in [15] which only re-optimizes over the set of time-overlapped reservations for a new incoming request. Although this approach may not give the results of an idealized global optimization strategy, it is still much more tractable and applicable to realistic networks. This framework is now presented further.

Consider the requisite notation first, as shown in Figure 1. The physical network is modeled as a graph, \(G(V, E)\), where \(V\) is the set of router switches and \(E\) is the set of physical links. All links \(e \in E\) have fixed capacity, \(C\), and their time-varying capacity levels are also tracked using a dynamic bandwidth-time function, \(c_e(t)\). Meanwhile VPN scheduling requests are denoted using a 5-tuple, i.e., \(r^n = (S^n, L^n, t^n_s, t^n_p, b^n)\), where \(n\) is the request index, \(S^n\) is the set of node sites (\(S^n \subseteq V\)), \(L^n\) is the set of virtual links between nodes in \(S^n\), \(t^n_s\) is the start time, \(t^n_p\) is the stop time, and \(b^n\) is the requested bandwidth, \(b^n \leq C\). Note that the above service model only requests bandwidth resources, although the formulation can be readily extended to include node-level resources (such as computing and storage). As per other ILP-based scheduling studies [15], it is assumed that time is discretized into fixed timeslots of duration \(T\), and all \(t^n_s\) and \(t^n_p\) times are integer multiples thereof. In addition, a host of other variables are also defined here:

- \(T_m\) is the maximum stop timeslot allowed in ILP mode
- \(v^n_i \in S^n\) is the \(i\)-th node selected in \(S^n\)
- \(L^n_{i,j}\) is the virtual link between \(v^n_i\) and \(v^n_j\) if \(v^n_i \neq v^n_j\)
- \(p^{n,e,k}\) is a binary flag which denotes overlay link occupation in time slot \(k\), i.e., \(p^{n,e,k}_{i,j} = 0\) if \(L^n_{i,j}\) does not use link \(e \in E\) at time slot \(k\); \(p^{n,e,k}_{i,j} = 1\) if \(L^n_{i,j}\) uses link \(e \in E\) at time slot \(k\)
- \(v \rightarrow e\) if \(v \in V\) is the egress node of link \(e \in E\); \(e \rightarrow v\) if \(v \in V\) is the ingress node of link \(e \in E\)
Using the above, the ILP objective function is defined as:

$$\text{minimize } \sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} \sum_{e \in E} \sum_{0 \leq k \leq T_m} b_e^n p_{i,j}^{n,e,k}$$  \hspace{1cm} (Eq.1)

subject to the following constraints:

$$\sum_{v_j^n \rightarrow e} p_{i,j}^{n,e,k} = 1 \quad t_s^n \leq k \leq t_e^n, v_i^n, v_j^n \in S^n$$  \hspace{1cm} (Eq.2)

$$\sum_{e \rightarrow v_j^n} p_{i,j}^{n,e,k} = 0 \quad t_s^n \leq k \leq t_e^n, v_i^n, v_j^n \in S^n$$  \hspace{1cm} (Eq.3)

$$\sum_{e \rightarrow v_j^n} p_{i,j}^{n,e,k} = 1 \quad t_s^n \leq k \leq t_e^n, v_i^n, v_j^n \in S^n$$  \hspace{1cm} (Eq.4)

$$\sum_{v_j^n \rightarrow e} p_{i,j}^{n,e,k} = 0 \quad t_s^n \leq k \leq t_e^n, v_i^n, v_j^n \in S^n$$  \hspace{1cm} (Eq.5)

$$\sum_{e \rightarrow v} \sum_{v \rightarrow e} p_{i,j}^{n,e,k} = \sum_{v \rightarrow e} p_{i,j}^{n,e,k} \quad t^n \leq k \leq t^n_{e'}, v \notin \{ v^n_i, v^n_j \}, v^n_i, v^n_j \in S^n$$  \hspace{1cm} (Eq.6)

$$\sum_{v_i^n \in S^n} \sum_{v_j^n \in S^n} b_e^n p_{i,j}^{n,e,k} \leq C \quad e \in E, 0 \leq k \leq T_m$$  \hspace{1cm} (Eq.7)

$$p_{i,j}^{n,e,k} = p_{i,j}^{n,e,k+1} \quad e \in E, t^n_s \leq k < t^n_e, v^n_i, v^n_j \in S^n$$  \hspace{1cm} (Eq.8)

Overall, the objective function in Eq. 1 tries to minimize the total network resource utilization. Meanwhile additional constraints are defined as well. For example, Eqs. 2 and 3 (Eqs. 4 and 5) ensure flow conservation at source (destination) nodes. Meanwhile, Eq. 6 ensures input/output flow conservation at transit nodes. Also, Eq. 7 limits the total provisioned bandwidth on a link to be below its maximum capacity, whereas Eq. 8 ensures route consistency during the requested time interval.

Now the authors in [7] also develop some basic heuristic schemes for VONS (VPN) scheduling. Although these offerings provide some initial solutions, there is clearly room for further improvement. In particular, scheduling re-routing strategies have been shown to give good gains for point-to-point connections, see Section II. Hence this concept is also leveraged here for VPN scheduling design.

An overview of the proposed re-routing scheme is shown in Figure 2. This algorithm follows a two-stage approach (as in [15]) where VPN re-routing or re-scheduling is triggered after failure of a regular attempt. Namely, initial VPN scheduling is done using any one of the basic VONS scheduling schemes in [7], i.e., either minimum hop or load-balancing. If this attempt fails, then the second stage is run to identify and re-route existing (scheduled) VPN requests to make room for the new request. One of the key objectives here is to carefully select the candidate subset of VPN demands for re-routing, i.e., to achieve a balance between computational complexity and blocking reduction. Note that each VPN request here is essentially treated as a set of connections pertaining to the VPN links. The overall pseudo code for the re-routing stage is shown in Figure 3 and now detailed further.

Overall, the re-routing heuristic first tries to route all overlay links in the request over a temporary copy of the network graph, \(G'(V, E)\), derived from \(G(V, E)\) by removing non-feasible physical links, i.e., without sufficient capacity in the request interval, \(c_e(t) < b^n\) in \([t^n_s, t^n_e]\) (steps 1-5, Figure 3). Note that this initial attempt can use any basic VPN scheduling algorithm from [7], i.e., hop-count resource minimization, load-balancing. If any link in the incoming VPN request cannot be routed, then the re-routing heuristic is triggered (steps 7-13, Figure 3). Namely, a candidate route is first computed for the failed link (step 7) using various strategies:

- **Minimum hop count** Compute shortest hop-count path between the link end-points using Dijkstra’s algorithm. This approach can yield saturation on specific links.
- **Minimum re-route number** Compute \(k\)-shortest paths (k-SP) between the link end-points, and for each path compute how many connections need to be re-routed to accommodate the link (see also Figure 4). Choose candidate route which requires the smallest number of re-routings. This approach tries to minimize complexity.
- **Threshold re-routing** Compute a candidate route which has at least a fraction \(\rho (0 < \rho < 1)\) of the requested link capacity. Namely, the bottleneck capacity of all links on the route, \(b^n_{\text{min}}\), must be greater than the fractional...
1: Given incoming request \( r^n = (S^n, L^n, t^n_s, t^n_e, b^n) \), generate temporary graph \( G'(V,E) = G(V,E) \)
2: Remove non-feasible links in \( G'(V,E) \), i.e., \( c_e(t) < b^n \) in \( [t^n_s, t^n_e] \).
3: for \( i = 1 \) to \( |L^n| \) do
4: if successfully compute a feasible path for \( i \)-th overlay link in \( L^n \) between \( t^n_s \) and \( t^n_e \) then
5: Reserve the path resources in \( G'(V,E) \) for the \( i \)-th link
6: else
7: Compute a candidate path for \( i \)-th link in \( L^n \)
8: Compute re-routing set
9: if Re-route candidate set succeed then
10: Reserve the candidate path resources in \( G'(V,E) \) for the \( i \)-th link
11: else
12: Re-routing process failed, Discard \( G'(V,E) \), Exit the loop, and Wait for the next request
13: end if
14: end if
15: end for
16: if all overlay link connections in \( L^n \) routed then
17: Setup successful, copy \( G'(V,E) \rightarrow G(V,E) \)
18: end if

Fig. 3. VPN demand re-routing heuristic

The actual available capacity into consideration.

Fig. 4. Sample overlapping link reservations and bandwidth availability

After computing the candidate route, the re-routing algorithm proceeds to identify a subset of (VPN) link connections to re-route, and these may impact multiple VPN demands (step 8, Figure 3). Namely, the scheme loops through all links along the candidate route, and for each one, tries to move a sufficient number of inactive scheduled reservations to free up enough capacity for the new link, i.e., if \( b'^{\min}_e < b' \) then the scheme iteratively removes inactive reservations \( r^n \) on link \( e \) with overlapping durations and re-computes \( b'^{\min}_e \) until \( b'^{\min}_e > b' \).

Finally, the algorithm tries to re-route each of the abovedescribed connections to ensure that no prior reservations are impacted (steps 9-13, Figure 3). To do this, another temporary graph \( G''(V,E) \) is generated by pruning the candidate route resources (computed in step 7, Figure 3). If all of the connections (identified for re-routing above) can be re-established over this modified graph, then the re-routing step is successful and the request is accepted. Otherwise the request is denied and the setup procedure (loop) is terminated. Note that additional improvements can also be devised here. For example, connection re-routing attempts can be ordered dynamically based upon increasing or decreasing capacity sizes. Such modifications are left for future study.

V. PERFORMANCE ANALYSIS

The performance of the ILP and heuristic re-routing schemes is now studied. Specialized simulation models are developed here using the OPNET Modeler environment. These models are further integrated with the CPLEX optimization tool to achieve “on-demand” ILP computations, i.e., by writing to an external ILP formulation file. All tests are done using the well-known NSFNET network topology. Here the network links are assigned 1 Gbps capacity each, and user-requested VPN sizes are varied between \( n = 4 \) to 6 nodes. Meanwhile, the associated VPN link requests are generated using the random Inet topology generator [16] to achieve a VPN node degree of about 2.5. All request book-ahead intervals, arrival intervals, and holding time are also exponentially-distributed.

Initial simulations are done to compare the performance of the dynamic ILP optimization with the various heuristic strategies. Namely, the hop-count without re-routing (HC-NR) scheme from [7] is chosen as a baseline option. In addition, several re-routing configurations are also tested, including hop-count with re-routing (HC-R), minimum-re-routing number re-routing (MR-R) and load-balancing with threshold re-routing (LB-R) with \( \rho = 0.5 \). Now carefully note that ILP scalability limitations prevent testing beyond a maximum lookahead time of about 50 seconds (timeslot size \( T=1 \) second). Hence the random inter-arrival times are scaled to have mean values of \( \lambda = 1 \) second, and the corresponding mean holding times \( \mu \) are adjusted as per desired load. In addition, the VPN link request sizes are varied uniformly from 2-10 Gbps. Finally, each test run is averaged over 1,000 random VPN requests, and a modified Erlang load metric is also defined to measure VPN input loads as follows:

\[
\text{Modified Erlang load} = \sum_{n=4}^{6} (n-1) \times \mu/\lambda \quad \text{(Eq.9)}
\]

where the overlay topology sizes range from \( n = 4 - 6 \) nodes and the \( 1/\lambda \) is the mean inter-arrival rate.
The overall blocking performance of the ILP and heuristic strategies is first shown in Figure 5. Now instead of measuring the number of failed VPN requests here, the bandwidth blocking rate (BBR) metric is used here, i.e., total failed requested bandwidth divided by the total requested bandwidth (across all VPN links). This value accounts for differing VPN sizes (in terms of link counts and bandwidth request sizes) and is a more accurate measure of bandwidth provisioning effectiveness. The overall BBR results indicate that the dynamic ILP scheme gives roughly the same BBR as the other re-routing heuristics. However, the non-re-routing HC-NR scheme gives about 5-10% higher blocking. In general, however, there is very little performance variability here due to the limited request size (due to ILP restrictions). Note that the ILP scheme can also be extended to re-optimize time-overlapping requests in conjunction with the new arriving request (i.e., since the model in Section III only considers an incoming request in isolation). This modification will likely lead to improved blocking performance and is left for future study.

Next, the average path lengths of successful reserved VPN link routes are also plotted in Figure 6. As expected, the ILP-based scheme gives the lowest resource utilization, followed by the HC-NR scheme. Meanwhile, the re-routing heuristics are generally more resource-intensive as they tend to yield longer detour routes to accommodate new demands. For example, the HC-R variant gives almost 10-15% higher usage values than its baseline HC-NR counterpart at medium-high loads.

Next, extended simulations are done to gauge the performance of the heuristic schemes only, i.e., without the limitations of the the ILP scheme. In particular, VPN link request are now varied from 200 Mbps to 1 Gbps (uniform) and mean request holding times are set to $\mu = 600$ seconds (exponential). The resulting BBR performance is shown in Figure 7, and the findings indicate about 19% lower blocking rates with the re-routing schemes as compared to the baseline HC-R scheme (log-scale plot). Moreover, there is very little observable difference between the re-routing strategies themselves. In addition, the MR-R scheme consistently gives lower blocking than the LB-R scheme for all tested loads and closely matches the HC-R scheme, i.e., under 2% difference in every case. The average path lengths of the successful VPN link routes are also plotted in Figure 8. Here there is no notable
difference between the performances of the various schemes, albeit the non-re-routing HC-NR scheme gives slightly lower (lowest) utilization values.

Finally, tests are also done to compare actual re-routing success rates for the re-routing heuristics, Figure 9. In general these results show that the LB-R scheme gives better success rates than the other two variants, especially under heavier load conditions. Note that different values of the re-routing threshold $\rho$ are also tested for the LB-R scheme (ranging from 0.1 to 0.9). However, these variations yield very little change in the overall performance results and hence are omitted here.

VI. CONCLUSIONS AND FUTURE WORK

Network virtualization scheduling allows operators to provide new innovative services for their clients. This work focuses on this area and presents a simplified optimization model to schedule user requests in an on-line manner. An advanced heuristic strategy is also proposed using re-routing strategies to improve request blocking. Results show that the optimization model gives slightly lower blocking and notably lower resource utilization, but can only scale to relatively small scenarios. Meanwhile the re-routing schemes give better performance as compared to some baseline virtual network scheduling heuristics. In particular, strategies which try to minimize the number of re-routed network links give the highest request acceptance rates. Future efforts will look at developing expanded optimization models to improve blocking reduction gains as well as service survivability concerns.

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


[16] Inet topology generator, see http://topology.eecs.umich.edu/inet/.