On the Mapping between Logical and Physical Topologies

Yun Hou[#], Murtaza Zafer*, Kang-won Lee*, Dinesh Verma*, Kin K. Leung[#]
[#]*EEE Dept., Imperial College, London, UK*
[*]*IBM T. J. Watson Research, NY, U.S.A.*
Email:[#]{yun.hou, kin.leung}@imperial.ac.uk, *{mzafer, kangwon, dverma}@us.ibm.com

***Abstract*— Network graphs, in general, successfully model a wide variety of interactions and relationships among entities, including both physical and logical connections. In this work, we study the problem of mapping a logical network on to a physical network; such a problem arises in various scenarios, for example, assignment of virtual machines on to physical servers in cloud computing, assignment of services on to physical devices in wireless/wire-line environments and physical resource assignment based on social networks. Specifically, in this paper, a logical network is a set of nodes with edges that denote the communication/bandwidth requirement between them, while a physical network denotes a set of physical nodes with edges that represent the available physical resources. The goal is to map the logical nodes on to the physical nodes and find physical resource allocation to meet the logical network demands, subject to physical network constraints. Towards this end, we propose a two-step approach to the problem, provide a set of novel feasibility checks for node assignment which are proved to be necessary and sufficient, and finally present a simple and fast algorithm that achieves a feasible logical to physical mapping with high probability. Illustrative simulation results are also presented to highlight the efficiency of the proposed algorithms.**

Keywords- Topology Mapping; Vitualization; Overlay Networks

I. INTRODUCTION

Generally, there are two approaches to deliver services in real life. One way is to buy new resources and build a new system according to the specific service requirements. The other way is to accomplish the required tasks using available resources in the existing system. The second approach can also be interpreted as a mapping problem between the logical plane and the underlying physical plane. As network graphs have been successfully used to model a wide variety of interactions and relationships among entities, we can use a logical graph to represent application specific requirements involving many entities and interactions between them and a physical graph to represent the actual physical resources available in the system. Given these two graphs, the problem that we study in this work is to generate a feasible mapping of the logical network onto the physical network, such that the logical requirements are met subject to physical network constraints. This abstract problem has applications in a wide variety of settings some of which are highlighted below.

In a cloud computing environment, a logical network could represent a set of virtual machines that must be deployed on to physical servers, where the logical links denote the communication requirements between them. The physical network consists of the physical servers and physical communication links between them. The above mapping problem then translates into assigning virtual machines onto physical servers and assigning flows in the physical network with bandwidth allocation to meet the logical communication requirements. In wireless (or wire-line) networks, nodes of a logical network could represent the various services (such as directory, routing and topology services) that must be deployed on some devices and these services need certain communication bandwidth between them for control purposes. The physical network here denotes the available physical nodes and communication links between them and the mapping problem is to provide a feasible mapping of the services onto devices with flow assignment such that the logical requirements are met. Finally within the context of social networks, a logical network could represent a social network denoting the various users and the amount of interactions between them. The mapping problem then is to assign physical resources to the various users to meet the communication requirements between them while also satisfying the physical network constraints.

In this paper, specifically, we consider the following setup. We consider two graphs, namely, the logical network and the physical network. Each graph is a set of nodes and edges where the edge value denotes the required bandwidth (demand) for the respective node pairs in the logical network, and for the physical network the edge value denotes the link capacity. The goal is to map each logical node onto one physical node, and to find flow assignment with bandwidth allocation on the paths, subject to physical network capacity constraints, such that the sum capacity of the flows is greater than the logical bandwidth requirement. Different from the graph isomorphic problems where one logical edge is directly mapped into a physical edge, we assume the logical edge, i.e., the requirements, can be realized by using multi-hop and multi-path physical routes.

Towards this end, we make the following notable contributions in this paper. We consider a 2-step approach to solve the problem, namely, 1) solve for a feasible node

assignment, and, 2) given the node assignment, solve for the flow assignment. The latter problem can be solved using multi-commodity flow algorithms [1, 2] for which a large set of literature exists. Hence, our focus will be on the node assignment problem which has received very little attention. We present a novel set of feasibility checks for a node assignment to be valid based on graph cuts; these conditions are analytically shown to be necessary and sufficient. Based on the graph cuts, we then present a simple and fast algorithm for node assignment with polynomial complexity that achieves a feasible mapping with high probability. Finally, we present illustrative simulation results highlighting the efficacy of various algorithms.

In distributed computing literature, the problem that we consider is related to the problem of task/job allocation to processors [3-7]; here the goal is to assign tasks to processors to optimize various different performance metrics. In [3, 4, 7], the formulation involves processing costs for each task and communication costs for flows and the goal is to minimize the entire program completion time. In [5], the sum of execution and communication costs is minimized for a homogeneous network, while in [6], a heterogeneous network model is considered. The main difference of this literature from our problem is that we do not consider any specific optimization but rather explicitly consider link capacity constraints on the physical network. We also address the important question of feasibility of mapping a given logical network onto a physical network by introducing feasibility checks based on logical-physical graph cuts. In another context in optical networks, the technology of wavelength division multiplexing (WDM) is utilized to construct logical networks from a given physical topology [8, 9]. However, here the node assignment is known a priori and the wavelength assignment is done to meet the logical demands. Finally, in overlay, peer-to-peer and virtual networking, the so-called diversified virtual network is mapped into the substrate network provider, i.e. mapping from logical to physical networks [10]. However, few existing works have addressed the feasibility issue of the mapping between two networks. In other words, whether or not a logical network can be mapped onto a physical network is still an open question.

The remainder of the paper is organized as follows. In Section II we formulate the mapping problem and decompose it into a Node-Assignment sub-problem and a Flow-Assignment sub-problem. In Section III a novel necessary and sufficient condition for the feasibility of the node-assignment problem is proposed and proved to be valid. Section IV elaborates a set of node-assignment schemes based on the feasibility condition and analyzes their complexities. Section V presents simulation results and finally Section VI concludes the paper.

II. PROBLEM FOMULATION

As mentioned in the introduction, we consider two network graphs, namely, a logical network and a physical network. The logical network is an undirected graph denoted as, $G_L = (V_L, X)$, where V_L and X are the logical node and edge set, respectively. Nodes in the logical network have a certain communication/bandwidth (demand) requirement, and this is denoted by the edge weight $X(I, J)$ from logical node I to node J. The logical nodes need to be mapped onto real physical nodes (computers/processors) within a given physical network. Let $G_P = (V_P, C)$ denote an undirected graph that represents the physical network, where V_P and C are the node set and the edge set, respectively. Let $C(i, j)$ denote the edge weight in the physical network, which represents the communication capacity of a physical link from node *i* to node *j* in the physical network. In general, one can also consider node capacities as well as mapping of multiple logical nodes into one physical node. However, for the sake of simplicity, we assume in this paper that each logical node must be mapped to a unique physical node. Therefore, the number of physical nodes, *N*, are taken larger than or equal to the number of logical nodes *M*.

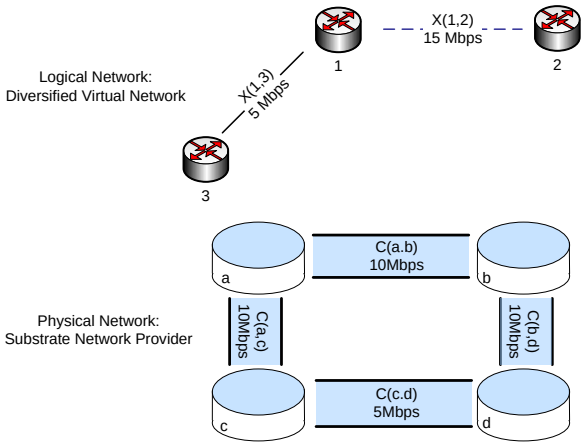

Figure 1 The logical and physical networks

An illustrative example of network virtualization showing a logical and physical network mapping is given in Figure 1 and Figure 2. The logical network, the diversified virtual network in virtual networking, is composed of three virtual routers (VR) 1, 2 and 3 with specified communication requirements among them, i.e., a bandwidth requirement of 15Mbps between VR 1 and VR 2 and 5Mbps between VR 1 and VR 3. The physical network, in virtualization networking, is referred to as the substrate network provider which is composed of four physical routers (PR) a, b, c, and d. The communication capacity between the PR's is given in the figure.

Then the mapping problem is to ask where VR 1, 2 and 3 should be implemented in the substrate network such that the bandwidth requirement between VR's can be guaranteed. In

this paper, we assume that one PR can accommodate at most one VR only and we allow multipath and multihop routes. Then, as illustrated by Figure 2, VR 1, 2, 3 are mapped onto PR a, c and d, respectively. Please note that since multipath and multihop routing are allowed, the flow requirement for the virtual link $X(1,2)$ is achieved by two paths with aggregated bandwidth of 15 Mbps in the substrate network. As a result, we say that the physical nodes a, c and d are mapped nodes. A physical node is unmapped if it is not mapped onto any logical node, such as node b in the physical network in Figure 2.

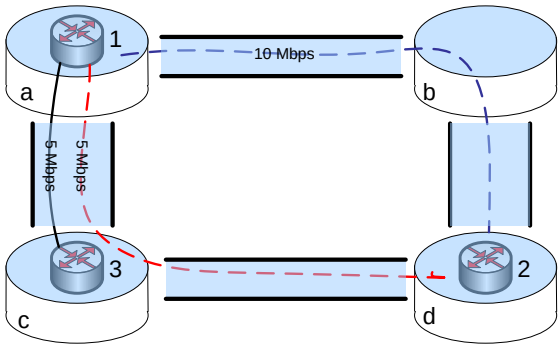

Figure 2 An example of mapping

Generally, given a logical and a physical network, the problem that we consider in this paper is how to map logical nodes into physical nodes such that the required communication between the logical nodes can be satisfied via multi-path and multi-hop flow routing, subject to the link capacity limits in the physical network. Mathematically, the problem can be formulated as follows.

Given a logical graph $G_L = (V_L, X)$ and a physical graph $G_P = (V_P, C)$, find (1) a node-assignment scheme, $\varphi(.)$, to map a logical node $I \in V_L$ into a physical node $i \in V_P$, i.e., $I = \varphi(i)$; and (2) find a flow-assignment scheme, i.e., a routing scheme which satisfies the capacity constraints in the physical network as well meets the logical requirements. The capacity constraints reflect the fact that the aggregated flow passing any one physical link cannot exceed the link capacity. Simultaneously, a feasible routing scheme must satisfy the requirement constraints, i.e. the actual allocated aggregated bandwidth on the set of flows between a node pair (i, j) in the physical network is larger than or equal to the logical demand requirement between the corresponding node pair $(\varphi(i), \varphi(j))$ in the logical network.

Consider a node pair (i, j) in the physical network and let there be a total of R_{ij} possible paths between them. Let $P_{ij} = \{P_{ij}(1), P_{ij}(2), \ldots, P_{ij}(R_{ij})\}$ denote the path set of flow (i, j) with $P_{ij}(l)$ representing the l-th path in the set P_{ij}. Let $\tilde{X}_{i,j}(l)$ be the allocated bandwidth for path $P_{ij}(l)$ and $\tilde{X}(i, j)$ denote the aggregated flow rate for flow (i, j), where $\tilde{X}(i, j) = \sum_{l=1}^{R_{ij}} \tilde{X}_{i,j}(l)$. Then, we can formulate the physical link capacity and logical demand requirement constraints as,

$$\sum_{i,j,l\,:\,P_{ij}(l) \subset (u,v)} \tilde{X}_{i,j}(l) \le C(u,v), \quad \forall u, v \in V_P, \tag{1}$$

$$\tilde{X}(i, j) \ge X\left(\varphi(i), \varphi(j)\right), \quad \forall \varphi(i), \varphi(j) \in V_L, \tag{2}$$

Given these two constraints, we say that a routing and flow assignment scheme is feasible if flows meet the constraints (1) and (2). Moreover, we say that a node-assignment is feasible if we can find at least one feasible routing scheme with it. In other words, if it is known that a node assignment is feasible, then it is known for sure that at least one feasible flow assignment can be found under that node assignment.

It is worth noting that given a particular logical-to-physical node mapping, the routing and bandwidth allocation problem is equivalent to the well-studied multi-commodity flow problem for which a rich literature exists. As a consequence, we can decouple the entire Node-Plus-Flow-Assignment (NPFA) problem into two sub-problems, namely, the Node-Assignment (NA) sub-problem where we find a feasible node mapping first, and then the Flow-Assignment (FA) sub-problem where we find the flows and bandwidth allocation for the assigned node mapping. Having this decoupled structure, we propose to take a 2-step approach to solve the entire NPFA problem by (1) first solving the NA problem and (2) solving the FA problem (using multi-commodity flow algorithms). By solving the NA problem first, one can also improve the success chance that the FA algorithms can finally find a flow assignment. The node assignment (NA) sub-problem will be the main challenge and focus of this paper.

One intuitive approach for finding a feasible node assignment is to exclude all the unfeasible node assignments by the use of feasibility checks. However, to the best of our knowledge very limited research effort has been put on the feasibility issue of the node assignment problem. In this paper, we obtain a new necessary and sufficient condition for the feasibility of node assignments which forms a set of feasibility checks to tell whether the given node assignment is feasible or not. Furthermore, using the graph cut methodology of feasibility checks, a fast and efficient 1-cut node assignment algorithm is derived as discussed in the subsequent sections.

III. THE FEASIBILITY CONDITION

In this section, we present the necessary and sufficient conditions for the feasibility of the node assignment problem. To proceed, we give some definitions first which will be utilized in describing the feasibility conditions.

Cut: As in standard graph theory, a cut is a set of edges that partitions the vertices of a graph into two sets [1]. Thus, here a cut refers to a set of links that separates nodes in the

physical network, including mapped and unmapped nodes, into two sets.

***n*-cut:** A *n*-cut is a cut that partitions the physical network into a set of n nodes and the other set of $N-n$ nodes, where $n \le N/2$. Note that for a network having N nodes, there are in total $C_N^n = \frac{1}{2}\binom{N}{n}$ possible *n*-cuts. Within this set, we denote a particular cut by $A_n(t), \quad \forall t = 1, 2, \ldots, C_N^n$.

Primary and complementary set: An *n*-cut partitions the physical network into two sets which we call as the primary and the complementary set, and denote them as $S_n(t)$ and $\bar{S}_n(t)$, respectively. The primary set is the one with smaller number of nodes.

Physical cut capacity: The physical cut capacity, denoted by $\gamma_n^P(t), \quad t = 1, 2, \ldots, C_N^n$, is the summation of the capacities of all the physical links in that cut, i.e.

$$\gamma_n^P(t) = \sum_{(i,j) \in A_n(t)} C(i,j), \quad \forall i, j \in V_P$$

Logical cut capacity: The logical cut capacity denoted by, $\gamma_n^L(t), \quad t = 1, 2, \ldots, C_N^n$, is the summation of the bandwidth requirements of all the logical links separated by that cut, i.e.

$$\gamma_n^L(t) = \sum_{i \in \mathbf{S}_n(t),\ j \in \bar{\mathbf{S}}_n(t)} X(\varphi(i), \varphi(j)), \quad \forall \varphi(i), \varphi(j) \in V_L$$

With the above notations, we now propose the following *n-cut* check.

***n*-cut check**: For a given node assignment, an *n*-cut check is to compare the *n*-cut capacity with the summation of the logical flows that go across the cut. An *n*-cut check is passed only if for all possible *n*-cut's (there are C_N^n *n*-cuts for an *N*-node network) the cut capacity is equal to or greater than the aggregated logical flows that go across the cut, i.e., the physical cut capacity is not smaller than the logical cut capacity. The *n*-cut feasibility check is mathematically written as:

$$\gamma_n^P(t) \ge \gamma_n^L(t), \quad \text{for all } t \in 1\{, 2, \ldots, C_N^n\}.$$

Given the above check, the feasibility condition for a node assignment can now be stated as follows:

Consider n-cut feasibility checks for n = 1, 2, …, $\lfloor N/2 \rfloor$. If all the checks are satisfied, the particular node assignment is feasible; vice-versa, if a particular node assignment is feasible, it must pass all the n-cut checks.

As an example to provide a graphical visualization for the checks, consider a 4-node logical network with vertices 1, 2, 3 and 4 that must be mapped onto a 4-node physical network with vertices a, b, c and d as shown in Figure 3 and Figure 4. Assume the node assignment is 1->a, 2->b, 3-> c, and 4->d. Then, for 1-cut checks, we compare the cut capacity of every 1-cut capacity in the physical network with the corresponding 1-cut capacity in the logical network. To obtain the 1-cut capacity in either the logical or the physical network, we just add up the edge weights associated with each node. As shown by the bold figures in Figure 3, the physical 1-cut capacities are 20, 15, 15, 20 associated with node a, b, c and d. Correspondingly, the logical 1-cut capacities are 20, 10, 10, 20 for the corresponding nodes (1, 2, 3 and 4respectively) in the logical network. Therefore, the 1-cut check is passed, since for all possible 1-cut's physical cut capacities are greater than their corresponding logical cut capacities.

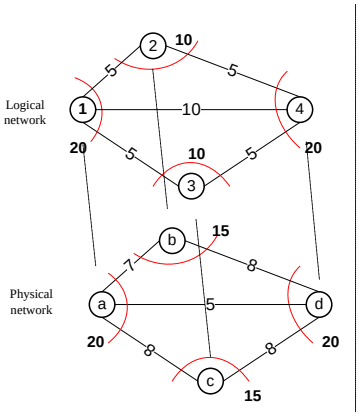

Figure 3 1-cut checks

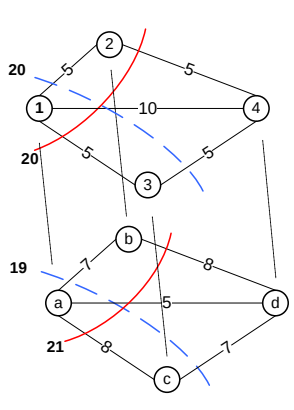

Figure 4 2-cut checks

Regarding the 2-cut capacities for this node assignment, we only review two out of three 2-cuts to illustrate how the graph cuts catch the unfeasible node assignments. One of the 2-cuts for the network is the cut that separates nodes 1 and 2 from nodes 3 and 4 in the physical network (separates nodes a and b from nodes c and d in the logical network). The cut is represented by the red solid arc in Figure 4. The logical cut capacity of this 2-cut equals 20, and the physical cut capacity of this cut is 21. Note we have to check all the possible 2-cut's to tell if the node assignment passes the 2-cut check. Then let us separate node 1 and 3 from node 2 and 4 in the physical network with the blue dashed cut in Figure 4. The logical capacity of this cut is 20, while the physical capacity of this cut is only 19. Since the logical cut capacity is greater than the physical cut capacity in this 2-cut, the node assignment does not pass the 2-cut check.

***Theorem 1**: If a node assignment passes all n-cut checks, for $n = 1, 2, \ldots, \lfloor N/2 \rfloor$, then the node assignment is feasible, i.e. one can find at least one feasible routing and flow assignment under this node assignment.*

Proof: We prove the correctness of Theorem 1 based on an equivalent statement of it. To proceed, we first present the equivalent statement and show its equivalence, followed by the correctness of the equivalent statement.

Assume a node assignment is done (regardless if it is feasible or not) and we run the following linear optimization problem to find the optimal routing and flow assignment for the objective function:

$$\max_{\tilde{\mathbf{x}}} \sum_{\substack{\varphi(i)\in V_L \\ \varphi(j)\in V_L}} \tilde{X}\left(\varphi(i),\varphi(j)\right) \tag{3}$$

$$s.t. \sum_{i,j,l\,:\,P_{ij}(l)\subset(u,v)} \tilde{X}_{i,j}(l) \le C(u,v),\ \forall C(u,v)\in C \tag{4}$$

$$\text{and } \tilde{X}\left(\varphi(i),\varphi(j)\right) \le X\left(\varphi(i),\varphi(j)\right),\ \forall \varphi(i),\varphi(j)\in V_L \tag{5}$$

where $\tilde{X}\left(\varphi(i),\varphi(j)\right) = \sum_{l:P_{ij}(l)\in P_{ij}} \tilde{X}_{i,j}(l)$ is the actual realized physical flow rate for the logical node pair $(\varphi(i),\varphi(j))$.

Next, we show that the two statements "there is at least one feasible routing under the node assignment" and "the optimal solution to (3) is

$$\tilde{X}^{*}\left(\varphi(i),\varphi(i)\right) = X\left(\varphi(i),\varphi(j)\right), \forall \varphi(i),\varphi(i)\in V_l$$

are equivalent to each other.

The reason for the equivalence between these two statements is as follows. If there is at least one feasible routing, then there must be a flow realization such that $\sum_{l:P_{ij}(l)\in P_{ij}} \tilde{X}_{i,j}(l) = X\left(\varphi(i),\varphi(j)\right)$ and this clearly must be a solution to (3) since it maximizes the objective function. In the reverse direction, if the optimal solution to (3) is $\tilde{X}^{*}\left(\varphi(i),\varphi(i)\right) = X\left(\varphi(i),\varphi(j)\right)$, then clearly all the logical requirements are satisfied and the solution to (3) gives the feasible routing scheme.

Having shown the equivalence of the statement, we now prove that if a node assignment passes all the n-cut checks for n=1,2,…, $\lfloor N/2 \rfloor$, the optimization in (3) must give an optimal solution where

$$\tilde{X}^{*}\left(\varphi(i),\varphi(i)\right) = X\left(\varphi(i),\varphi(j)\right), \forall \varphi(i),\varphi(j)\in V_L .$$

We will prove this statement by contradiction, where, we will show that if all the checks from 1-cut to *N*/2-cut are satisfied, it is not possible that the optimization in (3) has the optimal solution such that there is at least one logical node pair whose flow requirement is not satisfied.

Let us first make the contradictory assumptions as follows:

i) All the feasibility checks are passed;
ii) For an optimal solution of (3) there is at least one logical node pair for which the logical requirement is not satisfied.

We now show that assumptions i) and ii) cannot be true simultaneously. Let us put the unsatisfied node pairs into a set **U** such that

$$\begin{cases} \tilde{X}^{*}\left(\varphi(s_k),\varphi(t_k)\right) < X\left(\varphi(s_k),\varphi(t_k)\right), k\in \mathbf{U} \\ \tilde{X}^{*}\left(\varphi(s_k),\varphi(t_k)\right) = X\left(\varphi(s_k),\varphi(t_k)\right), k\notin \mathbf{U} \end{cases}. \tag{6}$$

Let us pick up an arbitrary unsatisfied node pair from set **U** and suppose $\tilde{X}^{*}\left(\varphi(s_k),\varphi(t_k)\right) = X\left(\varphi(s_k),\varphi(t_k)\right) - \Delta_k$ with Δ_k being strictly positive. Now, we find the min-cut [1] in the physical network that partitions node s_k and node t_k into two different sets. Let us refer to the primary set of the min-cut as $\mathbf{S}_m(l_{\min})$ and the complementary set of the min-cut as $\bar{\mathbf{S}}_m(l_{\min})$. Assume that s_k belongs to $\mathbf{S}_m(l_{\min})$ and t_k to $\bar{\mathbf{S}}_m(l_{\min})$. Further assume that there are m physical nodes in $\mathbf{S}_m(l_{\min})$ and $N-m$ physical nodes in $\bar{\mathbf{S}}_m(l_{\min})$ according to the min-cut partitioning. Then it is worth noting that this min-cut must have been checked using the *m*-cut feasibility condition, i.e., this min-cut which contains *m* nodes in $\mathbf{S}_m(l_{\min})$ is one of the *m*-cut's. Assumption i) then suggests

$$\gamma_m(l_{\min}) \ge \sum_{i\in\mathbf{S}_m(l_{\min}),\, j\in\bar{\mathbf{S}}_m(l_{\min})} X(\varphi(i),\varphi(j)), \tag{7}$$

where $\gamma_m(l_{\min})$ stands for the capacity of the min-cut with the cut size of *m*. Since $s_k \in S_m(l_{\min})$ and $t_k \in \bar{S}_m(l_{\min})$, the flow between $(\varphi(s_k),\varphi(t_k))$ goes across the min-cut. Thus we have

$$\begin{aligned} \gamma_m(l_{\min}) &\ge \sum_{\substack{i\in\mathbf{S}_m(l_{\min}) \\ j\in\bar{\mathbf{S}}_m(l_{\min})}} X(\varphi(i),\varphi(j)) \\ &\overset{(a)}{\ge} \sum_{\substack{i\in\mathbf{S}_m(l_{\min}) \\ j\in\bar{\mathbf{S}}_m(l_{\min}) \\ i\ne s_k,\, j\ne t_k}} \tilde{X}^{*}(\varphi(i),\varphi(j)) + \tilde{X}^{*}(\varphi(s_k),\varphi(t_k)) + \Delta_k \\ &\ge \sum_{\substack{i\in\mathbf{S}_m(l_{\min}) \\ j\in\bar{\mathbf{S}}_m(l_{\min})}} \tilde{X}^{*}(\varphi(i),\varphi(j)) + \Delta_k \end{aligned} \tag{8}$$

where, inequality (a) holds since for all the logical flows other than (s_k,t_k), the required logical flow are greater or equal to the actual achieved flows according to assumption ii). Next we introduce edge-load variable, $f(p,q)$, to denote the actual flows passing through a physical edge (*p*, *q*), due to the flow allocation from the optimization (3). Basically $f(p,q)$ is the aggregation of all the flow values of paths that use (*p q*). Thus for all the edges belonging to the set of the min-cut $A_{\min}(s_k,t_k)$, we have

$$\sum_{(p,q)\in A_{\min}(s_k,t_k)} f(p,q) = \sum_{\substack{i\in\mathbf{S}_m(l_{\min}) \\ j\in\bar{\mathbf{S}}_m(l_{\min})}} \tilde{X}^{*}(\varphi(i),\varphi(j)) \tag{9}$$

That is, the total amount of flows that go across the cut must be equal to the actual used capacity of all the links on that cut. This reflects the fact that every single path of a flow which goes across the cut must pass a link on the cut.

From (8) and (9) we have,

$$\gamma_m(l_{\min}) \geq \sum_{(p,q)\in \min-cut(s_k,t_k)} f(p,q) + \Delta_k , \qquad (10)$$

This means that there is at least Δ_k residual capacity available on the min-cut. According to the min-cut max-flow theorem [1], the maximum flow that is achievable between node s_k and node t_k must be $\sum_{(p,q)\in A_{\min}(s_k,t_k)} f(p,q) + \Delta_k$, i.e., we can add augmenting flows to deliver an additional amount of Δ_k from node s_k to node t_k. Thus, a solution to the optimization in (3) cannot have, $\tilde{X}^*(\varphi(s_k),\varphi(t_k)) < X(\varphi(s_k),\varphi(t_k))$, because the objective function has not been maximized. Hence, by contradiction, assumptions (i) and (ii) cannot be true which completes the proof. ■

Theorem 1 shows that the feasibility conditions given by the *n*-cut checks are sufficient for a node assignment to be feasible. We now show that they are necessary as well.

Theorem 2*: If a node assignment is feasible, then all n-cut checks, for* $n = 1, 2, \ldots, \lfloor N/2 \rfloor$, *must be satisfied.*

Proof: We prove the above statement by contradiction. Consider the following two assumptions:

a) The node assignment is feasible.
b) At least one n-cut check is violated.

Given (b), we put all cuts that have been violated into a set **D.** We then have

$$\gamma_m(l) < \sum_{i\in \mathbf{S}_m(l),\ j\in \bar{\mathbf{S}}_m(l)} X(\varphi(i),\varphi(j)), \quad \forall (m,l) \in \mathbf{D} . \qquad (11)$$

Similar to the proof of Theorem 1, we introduce $f(p,q)$ to denote the actual assigned flows on a physical edge from node *p* to node *q*. Then for all the edges belonging to the set of the cut, we have

$$\sum_{(p,q)\in A_m(l)} f(p,q) \leq \gamma_m(l) . \qquad (12)$$

However, due to the fact that every single path of a flow which goes across the cut must pass a link on the cut, we also have

$$\sum_{(p,q)\in A_m(s_k,t_k)} f(p,q) = \sum_{i\in \mathbf{S}_m(l),\ j\in \bar{\mathbf{S}}_m(l)} \tilde{X}(\varphi(i),\varphi(j)) . \qquad (13)$$

Putting (11), (12) and (13) together gives,

$$\sum_{\substack{i\in \mathbf{S}_m(l)\\ j\in \bar{\mathbf{S}}_m(l)}} \tilde{X}(\varphi(i),\varphi(j)) < \sum_{\substack{i\in \mathbf{S}_m(l)\\ j\in \bar{\mathbf{S}}_m(l)}} X(\varphi(i),\varphi(j)), \ \forall (m,l) \in \mathbf{D} \qquad (14)$$

From (14) we see that on a violated cut, the realized flow is less than the required logical flow, which is contradictory to assumption (a). Therefore, if a node assignment is feasible, it is certain that all feasibility checks must be satisfied. ■

IV. THE NODE-ASSIGNMENT ALGORITHMS

Given the n-cut feasibility conditions of the last section, a question that arises is how can we make use of them to find out feasible node assignments? This is the question that we answer in this section. Basically there are two approaches to make use of the feasibility checks to devise a node assignment (NA) algorithm. One way is to randomly pick up a node assignment and then run the feasibility checks, starting from the 1-cut check up to the N/2-cut check, to check if it is feasible or not. If the node assignment passes all the feasibility checks, we move on to step 2 to use multi-commodity flow algorithms to find the flow assignment. If the given node assignment fails to pass any of the checks, we stop and choose another random node assignment. This method is referred to as "The Random-Assignment-Plus-Check (RAPC)" algorithm. In the worst case, the RAPC needs to search all possible node assignments which could be N! in number for a N-node network. To avoid searching over all possible NA's, the other way is to devise an algorithm which intelligently picks up a feasible node assignment, and then run the flow assignment algorithm. In this approach, we propose an algorithm referred to as "1-cut-Mapping (1-cut MA)" algorithm.

A. The random-assignment-plus-check (RAPC) algorithm

The basic version of the RAPC approach is to pick up a random node assignment and check all the feasibility conditions from 1-cut up to *N*/2-cut for this node assignment until one of the feasibility conditions is violated or all the conditions are satisfied. We refer to this algorithm as the all-cut RAPC algorithm.

Next we analyze the expected complexity for the all-cut RAPC algorithm computed as the average amount of time that the all-cut RAPC spends to tell whether two networks can be mapped or not. In order to compute the average complexity, we define some probabilities first:

The m-cut catch probability: The *m*-cut catch probability, denoted by p_m, is the conditional probability that a node assignment passes all checks from 1-cut to (*m*-1)-cut, but fails to pass the *m*-cut check, given that the node assignment is unfeasible. In other words, this is the probability that the RAPC algorithm stops at the *m*-th cut.

The network-feasibility probability of two random networks: We say that a logical and a physical network are feasible, if there is at least one feasible node assignment for them. Then the **network-feasibility probability**, denoted by ε, is the probability that two randomly generated networks are feasible.

The NA-feasibility probability: This is the probability that a randomly picked node assignment is feasible, given that the logical network and the physical network are feasible. Let α denote the probability that a random node assignment is feasible for two feasible networks. Note the NA-feasibility

probability is associated with feasible networks only, since for unfeasible networks all the node assignments are unfeasible.

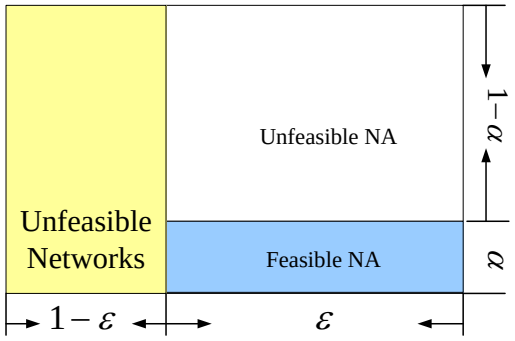

Figure 5 Probability illustration

We use Figure 5 to illustrate the dependency among situations corresponding to the above probabilities. Depending on whether the given networks are feasible or not, the complexity levels for the all-cut RAPC algorithm differ.

a) With probability $1-\varepsilon$, the given networks are not feasible. In this case, all the node assignment are unfeasible, hence the all-cut RAPC algorithm goes over all *N*! node assignments and with each node assignment the expected complexity is: $T \cdot \sum_{i=1}^{N/2} p_i C_N^i$, where *T* denotes the complexity to check an individual cut and $T \sim O(N^2)$.

b) With probability ε, the given networks are feasible. In this case, the all-cut RAPC algorithm goes over *k*-1 unfeasible node assignments with expected complexity $T \cdot \sum_{i=1}^{N/2} p_i C_N^i$ and then stop at the *k*-th node assignment which is feasible. For the *k*-th node assignment, because it is a feasible node assignment, the all-cut RAPC algorithm checks all feasibility conditions from 1-cut to *N*/2 cut. Then the complexity associated with the *k*-th node assignment is:

$$\sum_{m=1}^{N/2} T \cdot C_N^m = T \sum_{m=1}^{N/2} \frac{1}{2} \binom{N}{n} = T\, 2^{N-2}.$$

It is known that with the NA-feasibility probability of α, the expected value of *k*, $\bar{K}$, is $1/\alpha$.

Averaging over situations (a) and (b), we obtain the overall expected complexity for the all-cut algorithm:

$$(1-\varepsilon)\, N!\, T \sum_{i=1}^{N/2} p_i C_N^i + \varepsilon \left[\left(\frac{1}{\alpha} - 1 \right) T \sum_{i=1}^{N/2} p_i C_N^i + T\, 2^{N-2} \right] \quad (15)$$

The first term of (15) corresponds to situation (a), and the rest of the expression corresponds to situation (b). From (15) above, we see that the expected complexity for the all-cut RAPC algorithm is not polynomial. The high complexity is due to the NP-hard nature of the task-assignment problems [11]. As a result, we propose an alternative approach to only perform *m*-cut ($m < N/2$), feasibility checks for a random node assignment, and if all the checks from 1-cut to *m*-cut are passed, the algorithm identifies the given node assignment as feasible and enters the flow assignment phase. We refer to this approach as the ***m*-cut RAPC** algorithm.

Mathematically, the expected complexity of the *m*-cut RAPC algorithm can be computed as follows. The computations are very similar to what we did for the all-cut case to obtain (15). The expected value of *k* for the m-cut algorithm is:

$$\bar{K}_m = \frac{1}{\alpha + \left(1 - \sum_{j=1}^{m} p_j \right)}. \quad (16)$$

This is because the m-cut RAPC algorithm treats any NA that passes the m-cut check to be feasible, so the NA-Feasibility probability in the m-cut case becomes $\alpha + \left(1 - \sum_{j=1}^{m} p_j \right)$. Then, the expected complexity for the m-cut RAPC algorithm to tell whether two networks can be mapped or not is:

$$(1-\varepsilon)\, N!\, T \sum_{i=1}^{m} p_i C_N^i + \varepsilon \left[\left(\bar{K}_m - 1 \right) T \sum_{i=1}^{m} p_i C_N^i + T \sum_{i=1}^{m} C_N^i \right] \quad (17)$$

The simplest version of the *m*-cut RAPC category is the 1-cut RAPC where *m* = 1, which is with a complexity of

$$(1-\varepsilon)\, N!\, T p_1 N/2 + \varepsilon \left[\left(\bar{K}_m - 1 \right) T p_1 N/2 + TN/2 \right] \quad (18)$$

(18) shows that the complexity of the 1-cut RAPC algorithm is dramatically decreased to be polynomial for the case where the logical and the physical networks are feasible, compared with that of the all-cut RAPC algorithm. Obviously, the *m*-cut-RAPC trades in the certainty of feasibility for the speed of the algorithm. Since the *m*-cut algorithm cannot guarantee 100% catch of all unfeasible cases, we have to assess the effectiveness of this algorithm via the probability of erroneous decision.

Using the catch probabilities, we see that the probability that a node assignment is caught by the m-cut RAPC algorithm, given that the node assignment is not feasible. Then, the probability that an unfeasible node assignment cannot be caught by the m-cut RAPC algorithm is $1 - \sum_{i=1}^{m} p_i$.

An algorithm gives an erroneous decision when a node assignment is unfeasible but the algorithm identifies it as feasible, or when a node assignment is feasible the algorithm says it unfeasible. Our *m*-cut RAPC algorithm gives erroneous decisions only when the former situation occurs, because for a feasible node assignment the m-cut RAPC always gives the right answer. We refer to the probability of making erroneous decisions as the error probability, denoted as $p_e(m\text{-cut RAPC})$, and it is given as,

$$
\begin{aligned}
&p_e(m-cut\ \text{RAPC}) \\
&= \Pr\{\text{NA is unfeasible and NA passes n-cut}\} \\
&= \Pr\{\text{NA is unfeasible}\}\cdot\Pr\{\text{NA passes n-cut} \mid \text{NA is unfeasible}\} \\
&= (1-\alpha\cdot\varepsilon)\left(1-\sum_{i=1}^{m} p_i\right)
\end{aligned} \quad (19)
$$

An important observation drawn from (19) is that the tail distribution of the catch probability defines the effectiveness, i.e., the more trivial the tail probability mass is, the more effective the *m*-cut RAPC algorithm is. We will use simulations to investigate the tail probability mass of the catch probability with various depths *m* in Section V.

B. The 1-cut Mapping (1-cut MA) algorithm

The 1-cut MA algorithm is based on assigning nodes using their 1-cut capacities. Since only the 1-cuts are checked, as we show next, this algorithm has *ploynomial* complexity albeit at slightly higher error probability.

We first describe the 1-cut MA algorithm and then analyze its efficiency. The algorithm is composed of three steps, namely:

1) Sort the logical nodes in descending order in terms of their 1-cut logical capacity and re-label them in that order; i.e. $\beta_1(1) \ge \beta_1(2) \ge \cdots \ge \beta_1(m) \ge \cdots \ge \beta_1(M)$, where $\beta_1(m)$ represents the 1-cut capacity of the logical node labeled *m*.
2) Sort the physical nodes in descending order in terms of their 1-cut capacity and re-label them in that order; i.e. $\lambda_1(1) \ge \lambda_1(2) \ge \cdots \ge \lambda_1(n) \ge \cdots \ge \lambda_1(N)$, where $\lambda_1(n)$ represents the 1-cut capacity of the physical node labeled with *n*.
3) Starting from $i = 1$ and stopping until $i = M$, for each *i*, if $\lambda_1(i) \ge \beta_i(i)$, map the *i*-th logical node into the *i-th* logical node and repeat (3) with $i = i+1$; otherwise, stop the algorithm.

It is easy to see that if the above algorithm stops before $i = M$, no feasible mapping exists between the given logical and physical networks since the 1-cut feasibility check in all possible node assignment for the given networks would then be violated.

Next we analyze the complexity and effectiveness of this algorithm. Steps (1) and (2) are sorting algorithms which have complexity in the order of $O(N \log N)$. Step (3) is merely a simple compare operation. Therefore, the 1-cut MA node assignment is of $O(N \log N)$. Similar to the effectiveness evaluation in section III.A, we assess the effectiveness of 1-cut-MA via the probability of making erroneous decisions. Since every node assignment generated by the 1-cut-MA algorithm is guaranteed to pass the 1-cut check, the error probability for 1-cut MA is:

$$
\begin{aligned}
&p_e(\text{1-cut MA}) \\
&= \Pr\{\text{1-cut MA generates NA and NA is unfeasible}\} \\
&= \Pr\{\text{NA passes 1-cut check and NA is unfeasible}\} \\
&= (1-\alpha\cdot\varepsilon)(1-p_1)
\end{aligned} \quad (19)
$$

So the chance to give an error decisions for the 1-cut MA algorithm highly depends on the 1-cut catch probability. Simulation results presented in Section V show that for randomly generated logical and physical networks, the 1-cut catch probability is dominant as compared to *i*-cut catch probabilities for $i = 2, 3, \ldots, N/2$. Thus, it shows that the 1-cut MA algorithm is fast yet effective.

V. NUMERICAL RESULTS

In this section, we run simulations to obtain the catch probabilities to illustrate the effectiveness of the proposed NA algorithms. In order to get the catch probabilities, the logical network size, *M*, and the physical network size, *N*, are randomly chosen and both vary in the range of 2 to 10. For every combination of (*M*, *N*), the simulation randomly generates 10,000 logical and 10,000 physical graphs by assigning random edge weights to them. The edge weights are generated by the random number generator in MATLAB with uniform distribution $U(a_L, b_L)$ and $U(a_P, b_P)$ for logical and physical networks, respectively. The edge weights are generated with neither spatial nor temporal correlation. Thus, the weights of edges in the logical/physical network are i.i.d. In every snapshot, with a specific logical and a specific physical network, we first perform a totally random node assignment to place one logical node into one physical node, and then go through our checks starting from the 1-cut up to *N*/2-cut to check the feasibility of the assignment. As long as the *m*-th cut feasibility check is violated, the simulation counts the node assignment as unfeasible and stops the checking process for it. After that, the simulation enters the next snapshot to generate a new logical *M*-node network and a physical *N*-node network until 10,000 snapshots have been exhausted for the combination of (*M*, *N*).

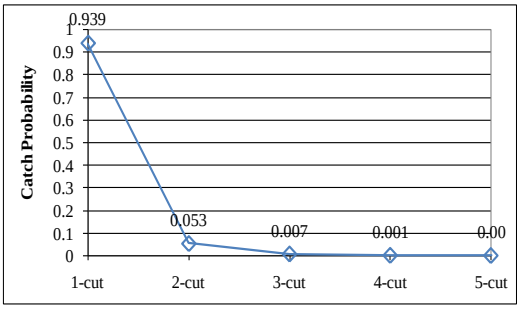

Figure 6 The catch probability for random networks with $a_L = 1, b_L = 10$ and $a_P = 1, b_P = 10$

One of the purposes of our simulation is to capture the characteristics of the catch probabilities. We first start with the case that the logical and the physical networks have identical

distribution for generating random edge weights. This corresponds to the scenarios where the requirements and the provision of resources are in the same range. Figure 6 shows the catch probability p_m for the proposed *m*-cut checks $m = 1, 2, \ldots, 5$ by averaging over all (*M*, *N*) combinations. It is noticeable that the 1-cut catch probability is 0.939 which means that by doing only 1-cut checks, we already catch 93.9% of the unfeasible assignments in this scenario.

Furthermore, Figure 6 confirms that the *m*-cut catch probability is a monotonously decreasing function of *m*, as expected in Section IV.*A*. Then a subsequent question is: if a node assignment passes all the checks from 1-cut up to m-cut, how much confidence do we have to say that this node assignment is feasible? To answer this question, we plot out the cumulative catch probability as a function of the depth of the feasibility check, as shown in Figure 7, to unfold the extent of certainty of the feasibility for a node assignment provided by the *m*-deep checks. A saturating trend of the cumulative catch probability is observed in Figure 7.

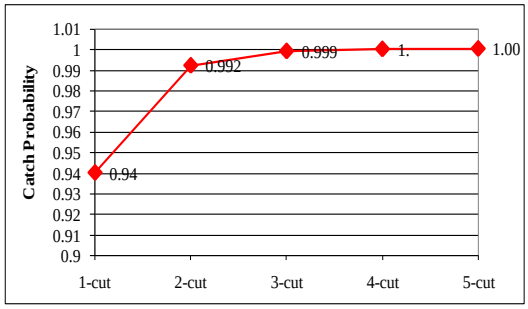

Figure 7 The cumulative probability of *m*-cut checks

As discussed in Section IV, the complexity of the RAPC NA algorithms grows at a rapid rate as the depth of the check increases, since the incremental complexity due to adding the *m*-th cut check is proportional to C_N^m . However, simulation results shown in Figure 7 suggest that the additional confidence of feasibility brought by running *m*-th cut feasibility check is decreasing as *m* increases. As a result, in *computational constrained* scenarios where computation resources are very expensive, our 1-cut MA algorithm which has the depth of check equal to 1 reaches a good balance between complexity and effectiveness.

We next provide simulation results for the cases when the edge weights of logical and physical networks vary in different ranges by using different mean values in the uniform distribution. The blue (inclined stripped) bars in

Figure 8 are the catch probabilities for the case that the logical demands are higher than the physical resources, statistically. The yellow (horizontally stripped) bars represent the catch probabilities in the situations where the demand in the logical network is lower than the physical resources on average. It is observed from Figure 8 that when the logical network requirement is statistically greater than the capacities in the physical network, the 1-cut catch probability is higher, compared to the cases that the demands are equal to or smaller than the resource supply on a per-edge basis. On the contrary, when the physical network has sufficient resource on every link, the 1-cut catch probability is slightly decreased.

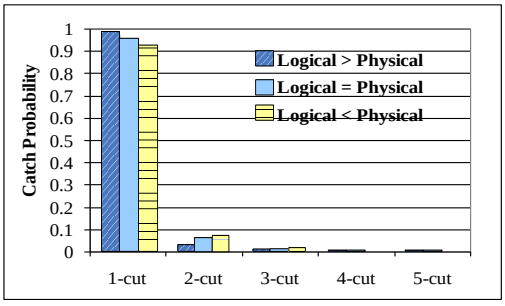

Figure 8 The catch probabilities for random networks.
"Logical > Physical": $a_L = 5, b_L = 14$ and $a_P = 1, b_P = 10$
"Logical > Physical": $a_L = 1, b_L = 10$ and $a_P = 1, b_P = 10$
"Logical > Physical": $a_L = 1, b_L = 10$ and $a_P = 5, b_P = 14$

Another interesting characteristic that can be captured by this set of simulation is the probability that a logical network and a physical network have a feasible node assignment, when the node assignment is randomly chosen. This is a quantitative measurement of how easily the RAPC algorithms can find a node assignment for randomly generated logical and physical networks. Therefore, we plot out the probability (the probability $\alpha \cdot \varepsilon$ in Section IV) of having a feasible random node assignment for different network size combinations in Figure 9.

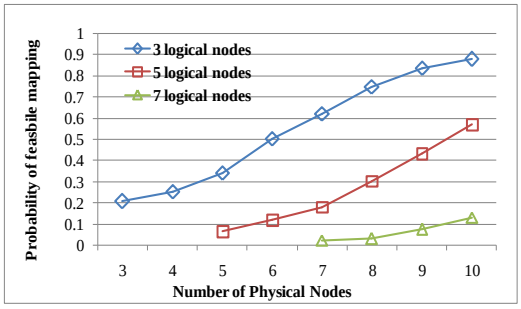

Figure 9 Probability that a (M, N) combination has feasible mapping

A very important and intuitive message conveyed by Figure 9 is that the increase of the probability of feasibility is proportional to the extent of the node redundancy in the physical network, when we fix the size of the logical network. This is unfolded by the increasing trend of all curves in Figure 9. In other words, it reflects the fact that when a physical network has a larger number of redundant nodes compared to the logical network, i.e., provides alternatives for routing; it is more likely to find a feasible node mapping to the logical network. A second message delivered by Figure 9 is that the benefit brought by the redundancy is inversely proportional to the size of the logical network. This is observed by comparing

the same *i*-th point among different curves in Figure 9. The *i*-th point on a curve stands for the chance of finding a feasible node assignment for the corresponding logical network when there are *i*-1 spare nodes in the physical network. In Figure 9 we find the *i*-th point on the 3-node curve is uniformly greater than the *i*-th point on the 5-node curve which is again always larger than that on the 7-node curve, for $i = 1, 2, 3 \text{ and } 4$.

Finally we pick up one specific (*M*, *N*) combination, (4, 4), to discuss the tradeoff between the complexity and the accuracy of the NA algorithms. We run a new set of simulations to obtain the Network-Feasibility probability and the NA-Feasibility probability for random networks of size (4, 4). Simulation results shows that the Network-Feasibility probability ε for (4, 4) is 0.3, and the NA-Feasibility α for (4, 4) is 0.46. With ε and α we compare two algorithms, namely, the all-cut RAPC and the 1-cut MA algorithm to highlight the tradeoff. According to (15), (17), (18) and (19), the complexity to make a decision and the probability of giving an erroneous decision is listed in Table 1.

TABLE 1: PERFORMANCE EVALUATION FOR (5, 5) AND (5, 10) COMBINATION.

		Complexity	Error Pr
(*N*, *M*) = (4, 4)	All-cut RAPC	566	0
	1-cut RAPC	523	5.3%
	1-cut MA	4	5.3%

Table 1 unfolds that the 1-cut MA node assignment algorithm is dramatically faster, compared with the RAPC algorithms. As discussed in Section IV, the complexity of the 1-cut MA algorithm is on the order of *N* while the RAPC algorithms are on the order of at least $N!N^2$. As discussed in Section IV, the high complexity of RAPC is intrinsic, because it is well-known that the node assignment sub-problem itself is NP-hard, if one wants to find an absolutely accurate answer. So we can treat the high complexity of the RAPC algorithms as a price for the 100% correctness. The question of choosing the RAPC algorithms or the 1-cut MA algorithm depends on the degree of error tolerance of the scenario in use. For error-tolerant scenarios, e.g., scenarios aiming at mapping as many as possible logical and physical graphs with limited computation resource or time, the 1-cut MA algorithm fits better. However, for scenarios that does not tolerate any error, the RAPC algorithms will be a reasonable choice. Another factor that affects the choice between these two algorithms is the complexity of the flow assignment problem. One of the negative consequences of making an erroneous decision is that FA algorithms is used but cannot find a feasible routing for the problem. So when solving the multi-commodity flow problem is expensive, it is not desirable to adopt the 1-cut MA algorithm to take the risk of paying the high price for nothing.

VI. CONCLUSIONS

In this paper, we investigated feasibility issues for the problem of mapping a logical network onto a physical network. We presented a novel set of feasibility checks for a node assignment to be valid based on graph cuts; these conditions are analytically shown to be necessary and sufficient. Based on the feasibility conditions, we then present a simple and fast algorithm for node assignment with polynomial complexity that achieves a feasible mapping with high probability; the algorithm is based on 1-cuts in the network. Finally, simulation results are presented which show that the proposed algorithm with low complexity is highly efficient.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper saddle River, NJ 1993.

[2] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, The MIT Press, Cambridge, MA, 2nd Edn. 2001.

[3] G. Attiya and Y. Hamam, "Task Allocation for Minimizing Programs Completion Time in Multicomputer Systems," *Lecture Notes in Computer Science*, vol. 3044/2004, pp. 97-106, 2004.

[4] G. Attiya and Y. Hamam, "Two Phase Algorithm for Load Balancing in Heterogeneous Distributed Systems", *In Proc. of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing* (EUROMICRO-PDP'04), 2004.

[5] C. Lee and K. G. Shin, "Optimal Task Assignment in Homogeneous Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 2, Feb. 1997.

[6] M. Kafil and I. Ahmad, "Optimal Task Assignment in Heterogeneous Distributed Computing Systems", *IEEE Concurrency*, vol. 6, issue 3, pp. 42-50, 1998.

[7] I. Ahmad, M. Dhodhi and A. Ghafoor, "Task Assignment in Distributed Computing Systems", *In Proc. of IEEE Conference on Computers and Communications*, Scottsdale, AZ, 1995.

[8] S. Banerjee and B. Mukherjee, "The Photonic Ring: Algorithms for Optimized Node Arrangements," *Journal of Fiber and Integrated Optics, spl. issue on Networking with Optical Technology*, vol. 12, pp. 133-171, 1993.

[9] A. Brzezinski and E. Modiano, "Dynamic Reconfiguration and Routing Algorithms for IP-over-WDM networks with Stochastic Traffic," *IEEE Journal of Lightwave Technology*, November, 2005.

[10] Yong Zhu, Mostafa Ammar "Algorithms for Assigning Substrate Network Resources to Virtual Network Components" INFOCOM 2006, Barcelona, Spain, April 2006.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to The Theory of NP-Completeness*, W. H. Freeman and Company, NY, 1979.