Autonomic Traffic Engineering for Network Robustness

Ali Tizghadam, *Member, IEEE,* and Alberto Leon-Garcia, *Fellow, IEEE*

***Abstract*—The continuously increasing complexity of communication networks and the increasing diversity and unpredictability of traffic demand has led to a consensus view that the automation of the management process is inevitable. Currently, network and service management techniques are mostly manual, requiring human intervention, and leading to slow response times, high costs, and customer dissatisfaction. In this paper we present AutoNet, a self-organizing management system for core networks where robustness to environmental changes, namely traffic shifts, topology changes, and community of interest is viewed as critical. A framework to design robust control strategies for autonomic networks is proposed. The requirements of the network are translated to graph-theoretic metrics and the management system attempts to automatically evolve to a stable and robust control point by optimizing these metrics. The management approach is inspired by ideas from evolutionary science where a metric, network criticality, measures the survival value or robustness of a particular network configuration. In our system, network criticality is a measure of the robustness of the network to environmental changes. The control system is designed to direct the evolution of the system state in the direction of increasing robustness. As an application of our framework, we propose a traffic engineering method in which different paths are ranked based on their robustness measure, and the best path is selected to route the flow. The choice of the path is in the direction of preserving the robustness of the network to the unforeseen changes in topology and traffic demands. Furthermore, we develop a method for capacity assignment to optimize the robustness of the network.**

***Index Terms*—Robustness, Graph Theory, Betweenness, Autonomic, Traffic Engineering, Markov Theory.**

I. Introduction

SINCE the inception of networked communication systems, network and system management has been crucial to ensure proper operation in regards to configuration, performance, fault, security, and accounting. Today, expert human resources and complex systems are required to control and manage an increasing plethora of networked devices and applications, ranging from small sensors to terabit routers. The explosion of the Internet and the proliferation of networked devices (peer-to-peer communications, grids, service overlay networks, sensor networks, mobile and wireless systems, etc.) create unique challenges for network and system management through highly dynamic and difficult to predict demand patterns. The complexity of networked systems and the cost of management are also constantly growing. Classical approaches to network management are not up to the task in this complex and dynamic environment.

Transport networks that can direct the traffic flows according to differentiated levels of QoS, availability requirements and price are key elements to generating revenue by enabling a rich offering of services and applications. In a general sense then, one of the principal challenges in network management today is to meet the service level agreement (SLA) requirements of different customers in the presence of highly unpredictable variations of fundamental network parameters. One can easily see that by improving the robustness of the network, the service availability for customers is also increased. There are three major types of variations that can affect the performance of the network: network topology and connectivity (including changes in capacity of the links); traffic demand matrix (the set of source-destination traffic flows), and community of interest (the set of active source-destination traffics). In this paper, we will call a network control strategy robust if it can accommodate uncertainties that result from changes in topology, traffic demand or community of interest.

The majority of traffic control systems in use by service providers are configured manually by human intervention. This leads to slow response times, high costs, and customer dissatisfaction. Furthermore, the continual growth in traffic volume, diversity, and heterogeneous requirements make it impossible to continue working with the present network management systems. Automated service and network management are essential to creating and maintaining a flexible and agile service/application delivery infrastructure that also has much lower operations expense than existing systems. There is now a consensus that future communication systems need to be autonomous, managing their own evolution, performance, fault, and security concerns without explicit user or administrator actions.

In the field of traffic engineering, which is the main focus of this paper, few automatic traffic management systems have been proposed in the literature, and most of these address only a part of the problem and leave other parts unattended. In this paper we focus on IP transport and we argue that the above traffic engineering system requirements can be met by a self-management system based on autonomic computing. We give an overview of the conceptual design of our autonomic traffic engineering system, but we focus on a set of essential graph theoretic algorithms that provide the means for adaptive management required by the autonomic system. This includes a traffic engineering algorithm to manage the flow of demands in the network as well as a weight assignment method to

Manuscript received ; revised .

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada (e-mail: ali.tizghadam, alberto.leongarcia@utoronto.ca).

Digital Object Identifier 10.1109/JSAC.2010.1001xx.

allocate optimal weights (capacity is an important special case) to the links so as to maximize the network robustness.

Our graph-theoretic approach is motivated by Darwin's theory of evolution, where every creature has a "survival value" quantifying its sensitivity to environmental changes. We propose a metric, network criticality, to quantify the survival value of a network with respect to changes in traffic, topology, and community of interest. We also formulate optimization problems that maximize the robustness of a network based on network criticality.

The paper is organized as follows. In the next section we consider prior related work. Section III presents an overview of our autonomic management system. Section IV presents a graph-theoretic analytical model for robustness which provides the basis for our autonomic algorithms. A robust routing plan (and flow assignment algorithm) is proposed in Section V and validated through simulation in Section VI. Conclusions are presented in Section VII.

II. Related Work

Adaptive system design typically involves striking a balance between optimal systems that achieve the best performance (at a cost of high complexity and sensitivity) and robust systems that achieve good performance (at lower complexity and sensitivity). Given the scale and diversity of current networks, it is not surprising that the preponderance of resource management systems opt for robustness [1], [2], [3], [4], [5]. In this paper we propose a graph theoretic metric for robustness that can be applied to network design. We then develop a family of algorithms that enable autonomic management through the optimization of this robustness metric.

Dekker and Colbert [6] investigate the robustness of network topologies using graph-theoretic concepts. They assess robustness according to the traffic levels induced in the network by node failures. They argue that "node connectivity" is the most useful metric in graph theory to study the robustness problem, and they examine the relationship between node connectivity and the degree of symmetry of the network and they suggest that it is important for robust networks to satisfy node similarity and optimal connectivity conditions. They also describe a number of ways to design robust networks that satisfy these conditions. In [7] the same authors introduce the symmetry ratio of a network. This metric is the ratio of the number of distinct eigenvalues of a network to its diameter. This metric is used to study the robustness of a network topology in the face of targeted attacks.

In [8] we present an approach for robust routing in core networks based on the notions of "link criticality" and "path criticality". Link criticality attempts to measure the impact of the failure of a particular link on the remainder of the network. Path criticality uses the criticality of the links in a path to measure its desirability from a robustness perspective. Our link criticality measure is inspired by the deterministic betweenness centrality for nodes in a graph [9]. For node k the betweenness centrality with respect to flows from source node i to destination node j is defined as the proportion of instances of the shortest paths from node i to j that traverse node k. The overall betweenness centrality of node k is the sum of the centralities over all source-destination pairs. Link betweenness is defined similarly.

In traffic management shortest paths are not necessarily the best path in all circumstances. For this reason in [8], we modify the notion of link criticality to consider all feasible paths. Let n_{ij} be the number of feasible paths between i and j, and let n_{ikj} be the number of paths between i, j containing the specific link k . The betweenness of node k for source i and destination j is then $\frac{n_{ikj}}{n_{ij}}$. The overall betweenness of link k is the sum of the betweennesses for link k over all i and j. Betweenness centrality provides a metric of how critical a link is in the network topology. Based on this metric, we proposed Path- Criticality Routing (PCR) [8] as a heuristic to select paths for a given flow in a manner that is robust to changes in traffic demand or network topology. Simulation results show that the PCR heuristic performs very well in a wide-range of network scenarios.

The success of the PCR heuristic convinced us that there must be a theoretical basis for its excellent performance, and that this basis must revolve around the notion of betweenness. Unfortunately the enumeration of paths does not lend itself to tractable analytic results that explain the behavior of PCR. However, we have found that the notion of Random Walk betweenness, introduced by Newman [10], do support the development of a rich set of tractable network optimization algorithms. In this paper we present the basic set of theoretical results that provide the foundation for our proposed traffic engineering system.

III. Conceptual Architecture of the Management System

The conceptual idea underlying our management architecture is inspired the theory of evolution. Evolutionary processes are good examples of self-organizing systems. Darwin's theory describes the process of natural selection by which each slight variation, if useful, is preserved [11]. Every process receives a survival value as a result of natural selection that quantifies its overall sensitivity or robustness to the external variations. In this paper we are looking for an appropriate survival value for communication networks. The survival value indicates how adaptable a system is to unexpected events [12].

Darwin's theory does not consider any "final target" for the evolutionary changes in the nature, but one can see that viewing survival as the goal can lead to an implicit optimization problem. Therefore we arrive at the view that the first goal of the management system is to keep the system alive under unforeseen circumstances. For our purposes, the system (network) can be modeled as a weighted graph, and our main service is data transfer.

In any network, from small designed networks, to large-scale social networks, and even to the Internet, connectivity is a crucial factor as it is essential for communication. Therefore, the first parameter to consider as a candidate for "survival value" is the connectivity of the graph. Any communication network should evolve in a way that maximizes the probability of future connectivity. This implies that the optimization must address the real-time efficiency and performance of the whole network as a short-term goal, while striving to maintain and improve the survival value of the network as a long-term goal.

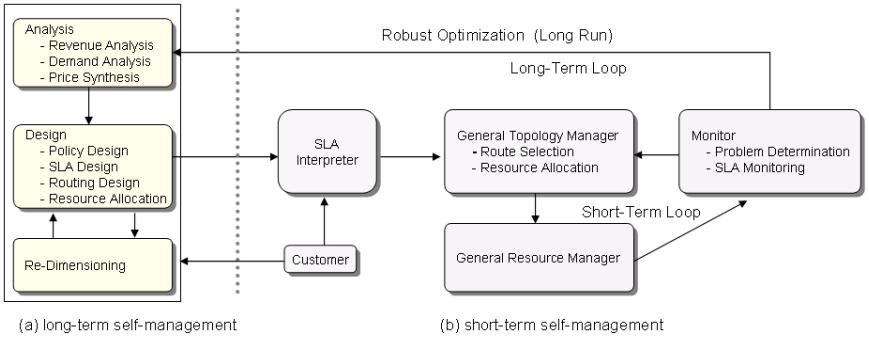


Fig. 1. Conceptual Architecture of AutoNet

To achieve long-term and short-term goals, one needs to have control mechanisms to monitor the present situation and make decisions accordingly using a corresponding controller. A simple model of such a system with two feedback loops, AutoNet, is shown in Fig. 1. The long-term loop in AutoNet learns a policy that has evolved as the result of gradual changes in the controlled system. In this evolutionary process, sometimes the control system cannot provide an appropriate policy. In this situation a re-planning process takes place and the new plan is put in place.

This slow/fast loop approach to the network management problem is the main building block of our management system. The main idea is to design appropriate long-term and short-term control loops to achieve the connectivity and performance simultaneously. Our performance metric should be the survival value of the network to directly reflect our goal, the optimization of robustness. The short-term part reacts to the network changes in real-time and the 'slow' part takes actions over a longer time-horizon.

The long-term loop develops the evolution based on an initial knowledge base that consists of the business policy as well as empirical results from previous experience about customer demand, network element reliability, price elasticity, etc. The network plan includes the translation of business policy into policies that are meaningful to the short-term part for use in the handling of customer requests. Our methodology is to convert the SLA to metrics from graph theory to capture both the topological aspects and SLA requirements. The plan also includes the synthesis of the SLA templates that will be offered to customers taking into account forecasted demand, resource requirements and price elasticity. All these planning parts are aimed at providing robustness through the long loop while providing immediate performance with the short-term loop. Finally, the plan also includes pre-partitioning of network resources to facilitate the handling of customer requests by the short-term part. For example, the plan may include pre-provisioned routes per each (ingress, egress) pair.

Because of the autonomic nature of the overall system, the short-term part needs to interact with the 'slow' (long-term) part when carrying out certain self-healing, self-optimizing, and self-configuring functions (bringing robustness as the final result). This mainly occurs when un-predictable events take place, such as sudden surges in demand or major failures in the network. In these situations the short-term part will respond to provide a fast real-time cure, but will act to provide a long-lasting cure by making a request for re-dimensioning to the 'slow' part. The interaction between slow and fast parts of AutoNet could also be the result of detecting inefficiency in resource usage in the fast part. In this case a request for re-dimensioning is sent to the slow part to re-optimize the allocation of resources.

The short-term or 'fast' part of the system consists of four major building blocks that are driven by customer requests. As shown in Fig. 1, the 'SLA Interpreter' block is responsible for negotiating the SLA with the customer and for converting the SLA contract to an appropriate form understandable by a 'General Topology Manager' block. This latter block plans the route and resource allocation based on the converted SLA, the already allocated resources, and current network demands. The results are delivered to the 'General Resource Manager' block which executes orders that allocate the appropriate amount of resources. The 'Monitoring' block continuously monitors the system to identify possible problems (e.g., SLA violations, failure alarms and so on). After filtering, it sends information to the 'General Topology Manager' to develop an immediate cure, and in parallel it may send a message to the 'Analysis' block of the 'slow' part to report an unpredictable event. If appropriate, the 'Analyze' block may initiate new network planning. In the next section we introduce the theoretical framework that enables the algorithms we propose for the General Topology Manager.

IV. Algorithmic Design of AutoNet

In this section we provide the theoretical results that form the basis for the design of the resource management algorithms (routing and flow assignment algorithm) in AutoNet (Fig. 1). To this end, we first develop a metric, ***network criticality***, to capture the robustness properties of a communication network. Then we use this metric to design appropriate algorithms for AutoNet.

A. Robustness and Network Criticality

To model the robustness in a network, one needs to consider the topology as well as the effect of load on the different nodes/links. In particular, the impact of a new flow on existing ones needs to be modeled. This motivates the use of betweenness metrics from graph theory. We consider the probabilistic definition of the node (link) betweenness as the main metric to quantify the criticality of a node or link and we use the

criticality metric to model the degree of robustness of the network.

In [10] a probabilistic interpretation of the betweenness is defined based on random walks in a graph. A random-walk starts from a source node s, chooses one of the neighbors at random with equal probabilities, and uses the link between source s and that neighbor to get there. The random walk continues wandering around until reaches at a specified destination d, where it stops. The betweenness $b_{sk}(d)$ of a node (link) k for source-destination pair $s-d$ is the expected number of times that a random walk passes node k in its journey from source s to destination d. The total betweenness of node k is the sum of this quantity over all possible $s-d$ pairs. We use a generalized definition of random-walk betweenness based on the weighted adjacency matrix of a graph. We consider a connected network which is shown by its graph G(N,E,W), where N, E, W are the set of nodes, links, and link weights of the graph respectively. Each node has a certain probability to send its data to the adjacent nodes. Let's assume a random walk at node s wants to go to node d as its final destination. Destination node is an absorbing state for this random walk and the walk is stopped in destination. The probability of passing node k in next step is shown by $p_{sk}(d)$ and defined as:

$$p_{sk}(d) = \begin{cases} 0 & if\ s=d \\ \frac{w_{sk}}{\sum_{q\in A(s)} w_{sq}} & otherwise \end{cases} \quad (1)$$

where $A(s)$ is the set of adjacent nodes of s and w_{sk} is the weight of link (s,k). The first condition in equation (1) is due to the fact that the destination node d is an absorbing node, and any random-walk coming to this state, will be absorbed or equivalently $p_{dk}(d) = 0$. Clearly, equation (1) defines a Markovian system.

Now, we define the node criticality for a weighted network simply as the random-walk betweenness of that node over the weight of the node.

$$\eta_k = \frac{b_k}{W_k}, \qquad W_k = \sum_{j\in A(k)} w_{kj}$$

where $\eta_k,\ b_k,\ W_k$ are the criticality, betweenness, and weight of node k (or weighted degree of the node) respectively. W_k is equal to the sum of all link weights incident to node k (weight of link (k,j) is shown by w_{kj}). Similarly, the link betweenness of link (i,j) (b_{ij})is defined as the expected number of times a random walk traverses the link summed over all source-destination pairs. The criticality of a link (i,j) (η_{ij}) is defined as the betweenness of the link over its weight:

$$\eta_{ij} = \frac{b_{ij}}{w_{ij}} \quad (2)$$

We will use criticality of the nodes (and links) to assess different networks based on their robustness to the changes in traffic demand, topology, and community of interest (source-destination pairs).

Observation 4.1: Equation (1) shows that if the weight increases, the goodness of that link (probability of being chosen) also increases. This means that for specific definition of weight, the QoS parameters which are in the direction of increasing the goodness should be positively related to the weight. We call these parameters "beneficial QoS parameters". In contrast, the QoS parameters for which increasing value denotes decreasing goodness, are called "detrimental QoS parameters". For example available bandwidth is a beneficial QoS parameter while used bandwidth or packet loss are detrimental QoS parameters.

Observation 4.1 suggests that SLA parameters can be mapped to the weights. In this paper we are interested in the study of the weight and its effect on robustness. We assume that SLA parameters are already mapped to the weights with an appropriate method. Some of these methods are discussed in [13]. This permits us to abstract different business policies and/or SLA's as parts of the weight definition. This is indeed the feature of an autonomic system that differentiates it from an adaptive mechanism [14].

One way of mapping QoS parameters to the link weights is as follows.

$$\begin{aligned} w_{ij} &= w_{ij}^{qos_1} \times w_{ij}^{qos_2} \times ... w_{ij}^{qos_k} \\ &= \frac{w_{ij}^{(1)}}{w_{ij}^{(a_1)}} \times \frac{w_{ij}^{(2)}}{w_{ij}^{(a_2)}} \times ... \frac{w_{ij}^{(k)}}{w_{ij}^{(a_k)}} \end{aligned} \quad (3)$$

where w_{ij}^{q} is a beneficial QoS parameter and $w_{ij}^{a_q}$ is a detrimental QoS parameter.

In order to find an expression for node betweenness we note that the path from any node i to j could be of length 0 to infinity. If we specify the probability values $p_{sk}(d)$ for destination d with matrix P_d, then for all $k \neq d$, the probability of entering node k at q^{th} step for different values of s and k can be obtained from corresponding entries of the matrix ${P_d}^q$ and in case of $k = d$ it would be 0. In our calculations, we treat the destination d as a fixed point and write all matrices based on this assumption. At the end we obtain the general results for our metrics by adding up the results for different destinations. One can write this relationship in matrix form as follows:

$$B_d = [b_{sk}]_d = \begin{cases} \sum_{q=0}^{\infty} P_d^q & if\ k\neq d \\ 0 & otherwise \end{cases} = \begin{cases} (I-P_d)^{-1} & if\ k\neq d \\ 0 & otherwise \end{cases} \quad (4)$$

where B_d is the betweenness matrix for destination d. By examining equation (4) one can easily see that the removal of column and row d from betweenness and probability matrices does not affect the other entries. We use $M(i|j)$ to denote the reduced matrix obtained by removing the i^{th} row and j^{th} column of matrix M. Equation (4) can be written as:

$$B_d(d|d) = (I - P_d(d|d))^{-1} \quad (5)$$

Let $W = [w_{ij}]$ be the weight matrix of the graph, D be the diagonal matrix of weighted degrees or graph nodes, and L be the Laplacian of the graph [15], [16]. We know that:

$$\begin{aligned} L &= D - W, \quad D = diag(W_1, W_2, ..., W_n) \\ P_d(d|d) &= D^{-1}(d|d) \times W(d|d) \end{aligned}$$

The last equation is the direct result of equation (1). Now we have:

$$\begin{aligned} I - P_d(d|d) &= I - D^{-1}(d|d) \times W(d|d) \\ I - P(d|d) &= D^{-1}(d|d) \times L(d|d) \end{aligned} \quad (6)$$

Replacing equation (6) in (5) results in:

$$B_d(d|d) = L^{-1}(d|d) \times D(d|d) \tag{7}$$

Note that the graph $G(N, E, W)$ is assumed to be connected which means that the rank of graph Laplacian L is $(n-1)$. As a result, the inverse of reduced Laplacian $L(d|d)$ exists and equation (7) has a unique solution. Now we need to write equation (7) in terms of the Laplacian of the original graph.

Lemma 4.2: For entries of the reduced inverse of the Laplacian matrix, one can write:

$$(L^{-1}(d|d))_{sk} = l^+_{sk} - l^+_{sd} - l^+_{dk} + l^+_{dd} \tag{8}$$

where l^+_{sk} shows the entry of row s and column k in Moore-Penrose inverse of Laplacian matrix L.

Proof: See Appendix A. ∎

According to the equations (7) and (8) , we can obtain the betweenness of the node k for source-destination pair $s-d$:

$$(B_d(d|d))_{sk} = (l^+_{sk} - l^+_{sd} - l^+_{dk} + l^+_{dd}) \times W_k$$
$$\frac{b_{sk}(d)}{W_k} = l^+_{sk} - l^+_{sd} - l^+_{dk} + l^+_{dd}$$

To obtain the total betweenness of node k, we need to consider the effect of all source-destination pairs.

$$\frac{b_k}{W_k} = \frac{1}{W_k}\sum_s\sum_d b_{sk}(d) = \frac{1}{W_k}\sum_s\sum_d \frac{b_{sk}(d) + b_{dk}(s)}{2}$$
$$= \sum_s\sum_d \frac{l^+_{dd} - l^+_{sd} - l^+_{ds} + l^+_{ss}}{2}$$

or:

$$\begin{aligned} \frac{b_k}{W_k} &= \frac{1}{2}\sum_s\sum_d (l^+_{ss} + l^+_{dd} - 2l^+_{sd}) \\ &= \frac{1}{2}\sum_s\sum_d \tau_{sd} = \frac{1}{2}\tau \\ \tau &= \sum_s\sum_d \tau_{sd} = \sum_s\sum_d (l^+_{ss} + l^+_{dd} - 2l^+_{sd}) \end{aligned} \tag{9}$$

To obtain equation (9) we used the fact that Laplacian matrix (and its Moore-Penrose inverse) is symmetric. It turns out that τ_{sd} is equal to the resistance distance between two nodes s and d [17].

A similar result can be derived for a link of the graph as well. For a link (i,j), one can find the betweenness of the link based on the betweenness of its two end nodes.

Lemma 4.3: Betweenness of link $l = (i, j)$ is equal to $b_{ij} = b(l) = \tau w_{ij}$. Equivalently, the criticality of link (i,j) would be $\eta_{ij} = \eta(l) = \frac{b_{ij}}{w_{ij}} = \tau$.

Proof: It is enough to note that $b_{ij} = b_i p_{ij} + b_j p_{ji}$, and apply equations (1) and (9). ∎

Observation 4.4: Equation (9) and lemma 4.3 show that the node/link criticality is independent of the choice of node/link.

Observation 4.4 is a significant result showing that the betweenness of a node (link) can be written as the product of two graph values ($b_k = W_k\frac{\tau}{2}$ for a node k or $b_{ij} = w_{ij}\tau$ for a link (i, j)) one of them is a local metric i.e. the weighted degree of a node (or weight of a link), and the other one a network-wide metric τ which is only a function of graph weight matrix. We call this global metric (τ) as *network criticality* and it will be our main tool to investigate the robustness of different networks. A smaller value of τ means a higher level of robustness. Indeed τ is the survival value that we need to model the robustness because it can be used to quantify the resistance of a network to the unwanted changes in network topology or traffic demands, the less the network criticality, the less the sensitivity to the changes in topology and traffic.

B. Network Criticality and Communication Networks

Network criticality can also be used to design methods to engineer the evolution of network flows or network topology. In the following we will show the importance of network criticality in the context of traffic engineering for communication networks.

Corollary 4.5: Let T be the average hop length of a random-walk (or average time that a random-walk is in the system) for all source-destination pairs, and B be the average node betweenness of all nodes. Then: $B = (n-1)T$.

Proof: See Appendix B. ∎

Corollary 4.6: The normalized betweenness of each node i of the graph is $\frac{b_i}{\sum_{k=1}^n b_k}$. It can be shown that this quantity is equal to the stationary probability of that node in a Markov chain built on the weights of the graph.

Proof: Equation (9) can be used to simplify the normalized betweenness of a node.

$$Normalized\ b_i = \frac{b_i}{\sum_{k=1}^n b_k} = \frac{\frac{1}{2}\tau W_i}{\frac{1}{2}\sum_{k=1}^n \tau W_k} = \frac{W_i}{W} = \pi_i$$

In these equations we have: $W_i = \sum_{j\in A(i)} w_{ij}$, $W = \sum_{i=1}^n \sum_{j\in A(i)} w_{ij}$. ∎

Now we are ready to investigate the relation between criticality and input traffic. Let λ be the average input rate at any individual node of the network, and let the weight of each link be the capacity of the link $(i, j) = l$ (i.e. $w_{ij} = c_{ij} = c(l)$). Further, let x_{max} be the average load on the node which has the maximum betweenness among all the nodes, and consider the capacity of this node as c^* . x_{max} can be approximated by the total average rate of this node times the average time that a demand is in the system.

$$x_{max} = n\lambda\pi_{max}T = \frac{\lambda}{n-1}b_{max}$$

To obtain this equation, we used corollary 4.5 and 4.6. Hence:

$$\begin{aligned} x_{max} &\leq c^* \Rightarrow \lambda \leq \frac{n-1}{\frac{b_{max}}{c^*}} \Rightarrow \lambda \leq \frac{2(n-1)}{\tau} \\ \max\lambda &= \frac{2(n-1)}{\min_W \tau} \end{aligned} \tag{10}$$

We used observation 4.4 to get the equation (10) . This result can be summarized in the following theorem.

Theorem 4.7: To maximize the capacity of a network, one needs to minimize the node/link criticality of the network.

Proof: This is a direct result of equation (10) and observation 4.4. ■

C. Optimization of Network Criticality

Now we should verify that the minimization of the network criticality is possible. To answer this question we need lemma 4.8.

Lemma 4.8: Network Criticality τ is equal to $2nTr(L^+)$.

Proof: Since $\tau_{sd} = l^+_{ss} + l^+_{dd} - 2l^+_{sd}$, we have

$$\begin{aligned}\tau &= \sum_{s,d}\tau_{sd} = \sum_d\sum_s l^+_{ss} + \sum_s\sum_d l^+_{dd} - 2\sum_s\sum_d l^+_{sd} \\ &= n\sum_s l^+_{ss} + n\sum_d l^+_{dd} - 2\times 0 = 2n\sum_i l^+_{ii} \\ &= 2nTr(L^+)\end{aligned}$$

■

The following theorem proves that the minimization of network criticality is in fact doable.

Theorem 4.9: τ is a strictly convex function of graph weights. Further, τ is a non-increasing function of link weights.

Proof: We note that function $f(X) = Tr(X^{-1})$ is strictly convex on X, if X is positive definite (see [18]). Therefore, considering well-known equation $L^+ = (L + \frac{J}{n})^{-1} - \frac{J}{n}$ [18] (J is an $n \times n$ matrix whose entries are all equal to 1),we can see that $\tau = 2nTr(L^+) = 2nTr(L + \frac{J}{n})^{-1} - 2n$ is strictly convex on matrix $L + \frac{J}{n}$ (since L is positive semi-definite, $L + \frac{J}{n}$ is always positive definite).

It is also not difficult to show that $\frac{\partial\tau}{\partial w_{ij}} = -2n\|L^+_i - L^+_j\|^2$, where L^+_i is the i^{th} column of L^+. This is always negative, therefore, τ is a monotone decreasing function of link weights. ■

We write the optimization problem to minimize τ), when there is a fixed budget for the link weights (the sum of all weights is fixed). the following theorem provides condition of optimality for the optimization problem.

Theorem 4.10: Consider the following optimization problem on graph G(N,E,W):

$$\begin{array}{cc} Minimize & \tau \\ Subject\ to & \sum_{(i,j)\in L} w_{ij} \le C \quad, C\ is\ fixed \\ & w_{ij} \ge 0 \end{array} \tag{11}$$

For the optimal weight set, W^*, we have:

$$C\frac{\partial\tau}{\partial w_{ij}} + \tau \ge 0 \quad \forall (i,j) \in E$$

.

Proof: See Appendix C. ■

We will also need the following lemma later in this paper.

Lemma 4.11: Network criticality is proportional to the sum of all link betweenness sensitivities, more precisely: $\tau = \frac{1}{m-1}\sum_{(i,j)\in E}\frac{\partial b_{ij}}{\partial w_{ij}}$, where m is the number of links of the network.

Proof: See Appendix D. ■

Mapping to our conceptual architecture, theorem 4.10 and lemma 4.11 provide foundation to build the major blocks of Fig. 1. We will develop methods and algorithms for network planning (long loop) as well as flow assignment (fast loop) for our conceptual architecture. This is the subject of the next section.

V. Design of the Fast and Slow Control Loops for AutoNet

We now discuss how the analytical results extracted in previous section can be used to design appropriate algorithms for long-term and short-term blocks of AutoNet. Theorem 4.10 and lemma 4.11 show the control mechanism that needs to be implemented to maximize the robustness. The evolution of the management state should be in the direction of minimizing the network criticality.

We first notice that the available capacity of a network is a key element in flow assignment problem. Clearly the paths with more available capacity are desired since the low available capacity paths are prone to congestion. Hence an intelligent routing plan should avoid routing the flows onto the low available capacity paths and should request for capacity increases for those paths if possible. In addition, the capacity planning phase should be done carefully to assign appropriate capacities to all the links (nodes) of the network. Therefore, *we define weight of link l=(i,j) as its "available capacity" to take into account the role of capacity.*

A. Slow Loop (Network Planning)

The main job of the slow loop in AutoNet is network planning. The goal is to find a set of link weights to minimize network criticality when the sum of all link weights is given. Therefore, the slow loop of AutoNet effectively solves optimization problem (11).

Optimization problem (11) can be converted to a semi-definite programming problem. Suppose $\Gamma = (L + \frac{J}{n})^{-1}$. In order to have a semi-definite program we need to have the constraints of the optimization as linear functions of semi-definite matrices. In fact Γ can be written as a semi-definite inequality. We consider matrix $\Theta = \begin{pmatrix}\Gamma & I \\ I & L+\frac{J}{n}\end{pmatrix}$. The necessary and sufficient condition for positive semi-definiteness of Θ is that its Schur complement [18] be positive semi-definite. The Schur complement of Θ is $\Gamma - (L + \frac{J}{n})^{-1}$.

$$\Theta = \begin{pmatrix}\Gamma & I \\ I & L+\frac{J}{n}\end{pmatrix} \succeq 0 \Leftrightarrow \Gamma \succeq (L + \frac{J}{n})^{-1} \tag{12}$$

where $\succeq$ means positive semi-definite. Since the optimization problem (11) should minimize $Tr(\Gamma)$, the equality in equation (12) is chosen which is equal to $\Gamma = (L + \frac{J}{n})^{-1}$.

Now optimization problem (11) can be converted to a semi-definite programming.

$$\begin{aligned} Minimize \quad & 2nTr(\Gamma) - 2n \\ Subject\ to \quad & Diag(Vec(W)).\overrightarrow{1} = C \\ & \begin{pmatrix} \Gamma & I \\ I & L + \frac{J}{n} \end{pmatrix} \succeq 0 \\ & Diag(Vec(W)) \succeq 0 \end{aligned} \tag{13}$$

We have changed the first constraint of optimization problem (11) to an equality, since the optimal answer is the same. Also, note that $Diag(Vec(W))$ is a diagonal matrix with w_{ij}'s in main diagonal. This matrix is positive semi-definite because $w_{ij} \geq 0 \ \ \forall (i,j) \in E$.

This new optimization problem can be solved with standard methods of solving semi-definite programs. There are also some software tools to solve semi-definite programs. The slow loop of AutoNet implements this semi-definite program to plan a robust network. This network planning can happen at the initial step, or in the middle of traffic engineering activities due to request from fast loop.

An important special case of network planning is the capacity assignment problem. Consider a network $G(N, E, W)$ where the link weights are equal to the link capacities (when network is not loaded the available capacity of a link is equal to its total capacity), that is, $w_{ij} = c_{ij} \ \forall (i,j) \in E$ (c_{ij} denotes the capacity of link (i,j)). We investigate the capacity assignment problem in which network topology and link traffic loads $\gamma_{ij} \ \forall (i,j) \in E$ are assumed known and fixed. The goal is to find the capacity of the links so as to minimize the network criticality under the constraint that the total cost of the planning is fixed. The optimization problem remains the same unless the constraint set $w_{ij} \geq 0$ which is converted to $c_{ij} \geq \gamma_{ij}$. By applying the change of variable $w_{ij} = c_{ij} = c'_{ij} + \gamma_{ij}$ and $C' = C - \sum_{(i,j)\in E} \gamma_{ij}$ to the optimization problem (11), we will have the following convex optimization problem for capacity assignment problem.

$$\begin{aligned} & Minimize \quad \tau \\ Subject\ to \quad & \textstyle\sum_{(i,j)\in E} c'_{ij} = C' \quad , C' \ is \ fixed \\ & c'_{ij} \geq 0 \end{aligned} \tag{14}$$

B. Fast Loop (Traffic Engineering)

The main building block of the fast loop of AutoNet is the "General Topology Manager" (Fig. 1). For this block, we need to design a robust routing scheme that is able to cope with unpredicted changes in traffic and topology.

The control loop should always keep the present value of network criticality, and compare it with a reference value, in order to provide necessary input for the controller to make appropriate decisions.

A simple diagram of the control loop for traffic engineering purposes in AutoNet, including fast and slow loops, is shown in Fig. 2. The main idea is to find appropriate paths to run the flow so that the change in network criticality is minimized. Suppose a demand for source-destination pair $S - D$ needs to be routed and there are three eligible paths between node S and D. The controller should choose the path which creates the least network criticality after the demand is serviced. We design our controller (or traffic engineering block) based on this philosophy. Lemma 4.11 provides us with an appropriate approach. According to lemma 4.11, the network criticality is proportional to the sum of link betweenness sensitivities. Therefore, in order to minimize the network criticality, one should minimize the sum of link betweenness sensitivities. This suggests the choice of link betweenness sensitivity as the cost of a link. For a link l=(i,j):

$$cost(l) = cost(i,j) = \frac{\partial b_{ij}}{\partial w_{ij}} \tag{15}$$

Now it is enough to apply the Dijkstra's algorithm to find the shortest path(s) between every two nodes. This will find the paths that have the minimum impact on criticality. In case of more than one shortest path the one that causes less changes in network criticality, will be chosen.

It is worth mentioning that the dynamics of the network are summarized in the random-walk link betweenness, and by using the value of random-walk link betweenness we in fact apply a kind of implicit control law in the form of a policy. According to this policy, if the betweenness of a link increases, the risk of using the link also increases and the traffic engineering block tries to find a path which does not include this risky link. In other words, an adaptive control mechanism in implicitly used in the fast control loop.

The explicit control error signal in Fig. 2 determines whether the demand can be accepted. Suppose the initial value of network criticality when the network is not loaded is τ_{ini}. We accept the new demand only if there will be a path with enough available capacity and if the new network criticality after running the demand won't be more than $tr \times \tau_{ini}$, where tr is a threshold factor. Therefore, the reference value of our control loop is $\tau_{ref} = tr \times \tau_{ini}$. The choice of tr depends on the level of accuracy and robustness that we need. We found that $tr = 4$ works well in our tests. Note that according to lemma 4.11 network criticality is a monotone decreasing function of link weights, therefore, as long as we do not add to the initial capacity of the links (by network replanning via slow loop), the initial value of the network criticality is the minimum one and decreasing the available capacity of a link (i.e. link weight) will increase the value of τ, hence τ_{ini} can be used as the reference.

In this paper we use a binary control error as the input to the controller (traffic engineering block). We reject a flow and possibly ask for replanning if there won't be any path to guarantee $\tau_{ref} - \tau \geq 0$, otherwise we select the path and send it to the resource manager block for resource allocation. This process provides a simple call admission control mechanism for the traffic engineering block. In the resource block all the parameters will be updated and new values of network criticality as well as link parameters including weight, betweennesses, and betweenness sensitivities will be approximated.

To approximate the time complexity of the algorithm, we note that the running time to get the Moore-Penrose inverse is $O(mn^{\frac{1}{2}})$ [18], where m and n are the number of links and nodes in the graph respectively (we need to have the Moore-Penrose inverse of Laplacian matrix to find betweenness of the links and network criticality). The main part of the traffic

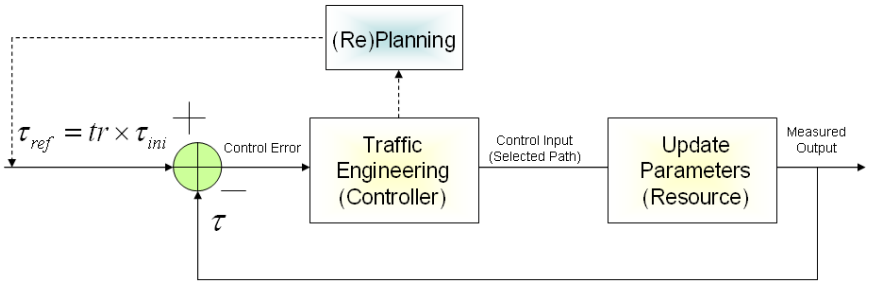
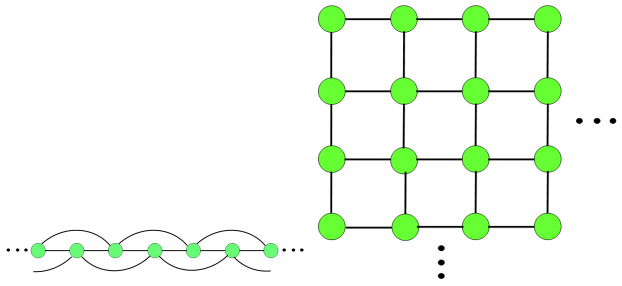
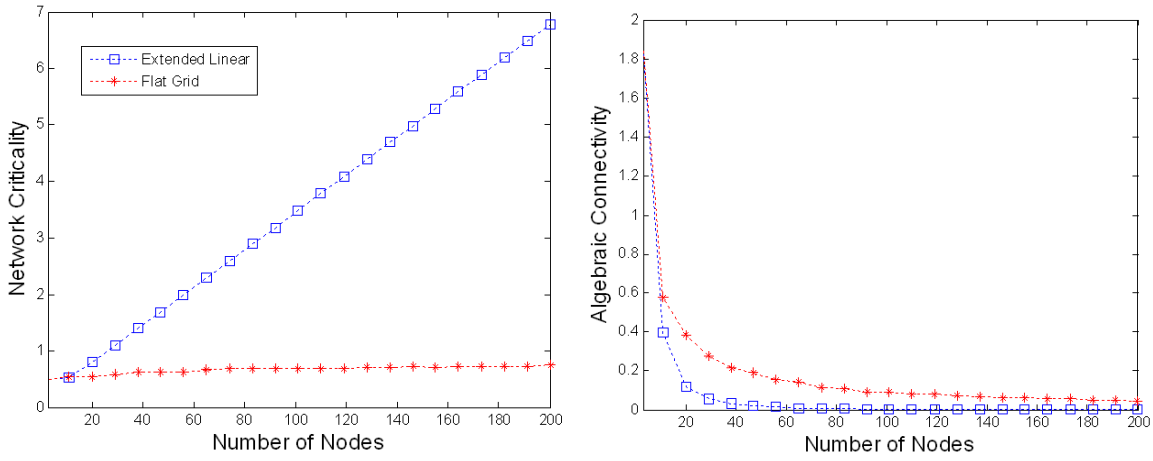


Fig. 2. Autonomic Loop for Traffic Engineering



(a) ELG (b) FGG



(c) Variations of τ (d) Variations of λ_2

Fig. 3. Behavior of Network Criticality and Algebraic Connectivity for ELG and FGG

engineering block can be obtained in $O(nlog(n))$ as it is just a shortest path algorithm with link costs. Hence the complexity of the algorithm would be $O(mn^{\frac{3}{2}}log(n))$.

VI. Evaluation

In this section we conduct some experiments to show the validity of AutoNet blocks. We start with a discussion on the importance of network criticality by comparing τ with algebraic connectivity which is another metric to measure the robustness of a graph.

A. Network Criticality and Algebraic Connectivity

Fiedler [19] defined algebraic connectivity as the first non-zero eigenvalue (λ_2) of the Laplacian matrix of a connected graph (the first eigenvalue of Laplacian matrix for a connected graph is zero). Algebraic connectivity is a lower bound for node connectivity and link connectivity. Therefore, increasing λ_2 will improve the connectivity of a graph.

In first experiment we consider extended linear graph (ELG), and flat grid graph (FGG) (see Fig. 3)and we compare the behavior of network criticality and algebraic connectivity in ELG and FGG for different network sizes. In this experiment we assume all the link weights are equal to 1. Fig. 3-(c) shows the behavior of network criticality for ELG and FGG. The criticality of ELG grows much faster than FGG. Fig. 3-(d) reveals that the algebraic connectivity of FGG is always better than ELG, that is, the flat grid has better connectivity but the speed of decreasing the connectivity of the graph is much slower than increasing the network criticality.

The main finding of this experiment is that while the changes in algebraic connectivity of ELG and FGG are relatively similar, there is a huge change in the behavior of network criticality, which means that network criticality captures some robustness properties of the graph that cannot be found in algebraic connectivity. Indeed this experiment reveals that increasing topological dimension of a network will increase its

TABLE I
CAPACITY ASSIGNMENT AND LINK DELAY USING 3 DIFFERENT METHODS

Link	Load	Kl(C)	Me(C)	Cr(C)	Kl(D)	Me(D)	Cr(D)
1	3.15	27.93	27.00	29.63	40.36	41.93	37.76
2	3.55	29.85	27.40	33.31	38.02	41.93	33.60
3	0.13	5.16	23.98	12.67	198.67	41.93	79.71
4	3.64	30.28	27.49	32.95	37.54	41.93	34.12
5	0.82	13.46	24.67	13.36	79.10	41.93	79.71
6	3.88	31.38	27.73	33.64	36.36	41.93	33.60
7	9.95	53.99	33.80	36.43	22.71	41.93	37.76

TABLE II
AVERAGE NETWORK DELAY AND NETWORK CRITICALITY USING DIFFERENT METHODS

Method	Average Network Delay	Network Criticality
Kleinrock	44.72	1.06
Meister	55.01	0.80
Criticality Method	49.30	0.56

robustness. Note that FGG expands in two dimensions whereas ELG grows in one dimension. A more detailed comparison of network criticality and algebraic connectivity can be found in [20].

B. Network Planning (Slow Loop)

In the following example our proposed optimal weight assignment method for long loop of AutoNet is compared with Kleinrock's method for capacity assignment [21], [22] and Meister's extension [23]. We use the telegraph network from Kleinrock's book (see [21], pp. 22-23).

Kleinrock's method finds capacities of the links in such a way to minimize the average delay of the network under the independence assumption and when the link loads are known. One problem with Kleinrock's approach is that it assigns very long delays to the links with small loads. Meister's method is an alternative approach which assigns equal delays to all the links, of course at the expense of a large deviation from optimal average network delay that can be achieved by Kleinrock's solution.

The proposed solution in this paper assigns capacity of the links in a way to balance the individual link delays so as to have acceptable link delays while still we have a good average network delay. Table I shows the capacity assigned to the links using all the methods. The second column of table I shows the individual link loads. Columns 3, 4, and 5 show the optimal capacity assignment using Kleinrock's method, Meister's method, and our proposed method (which we call criticality method) respectively. The minimum average network delay for these methods are given in second column of table II. The third column also shows the value of network criticality. In the criticality method we actually optimize the robustness (not the average delay as it is the case in Kleinrock's and Meister's method), therefore it is not surprising to see that the average delay obtained by criticality method is between two extremes of Kleinrock (to minimize the average network delay) and Meister (to minimize the maximum link delay).

Columns 6, 7, and 8 of table I show individual link delays for three methods. Kleinrock's method assigns very large delay to link 3 because the demand on link 3 is much less than the other links. Meister's method assigns equal delays to all the links. This resolves the issue with Kleinrock's method, but introduces a fairness problem. In our proposed method, the link delays are not equal to allow for fairness based on the demand for each link, and at the same time the individual link delay are kept in a reasonable range.

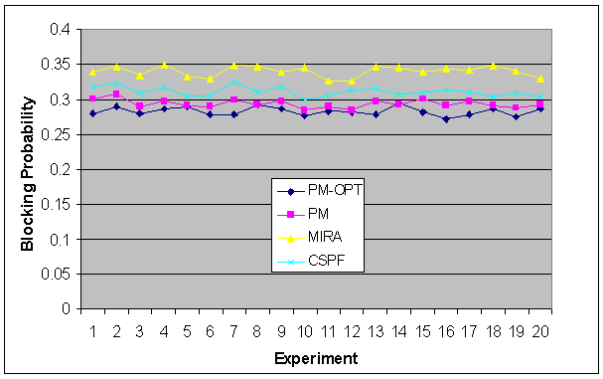


Fig. 4. Result of applying Proposed Method (PM) (capacities are not optimal), Proposed Method when capacities are optimal (PM-OPT), MIRA, and CSPF to the network under test

C. Traffic Engineering (Fast Loop)

In order to investigate the effectiveness of our traffic engineering algorithm, we ran a set of simulations on an anonymized version of a real network whose topology and sample traffic matrices are given in [24]. We apply our traffic engineering method to create LSPs (Label Switch Path) assuming that MPLS is used in the network to create the paths.

In the first experiment the requests for LSP arrive between each source-destination point (which is chosen at random) according to a Poisson process with an average rate λ, and the holding times are exponentially distributed with mean μ. We set $\frac{\lambda}{\mu} = 1800$ in our experiments (this provides a "heavily loaded" scenario in our test). We generate 10000 requests and measure the rejections or blocking for each one of the algorithms. In our tests the bandwidth requests for paths (LSPs) are taken to be uniformly distributed between 1 to 3 units. In Fig. 4 we show the blocking probability and compare the performance of our proposed method (PM) with Minimum Interference Routing Algorithm (MIRA) [1], a well-known MPLS path setup algorithm, and Constrained Shortest Path First (CSPF) [25], where the constraint in our case is the minimum bandwidth required per link . The test is performed 20 times and each time with 10000 path requests. One can easily see that the proposed method has the best performance, the CSPF is in second place and MIRA is in next position.

To investigate the effect of slow loop, we conducted another set of experiments. We found the optimal capacity assignment to minimize the network criticality by solving the optimization problem (13), and then repeated the same set of experiments for dynamic traffic. The results of this part are also shown in Fig. 4 (PM-OPT) . According to the results of this last experiment, we observe that starting with optimal capacities pre-planned in slow loop of AutoNet decreases the blocking probability of our proposed traffic engineering method.

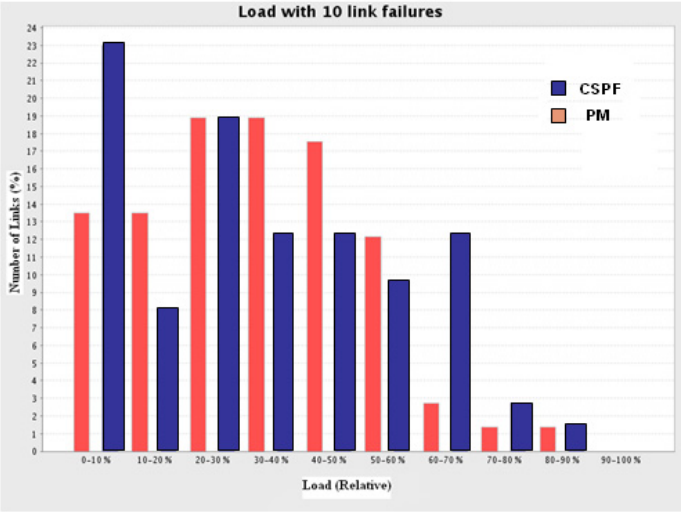


Fig. 5. Failure Test: Load distribution on different links while failure happens, comparing our method with CSPF

In order to study the effect of link failure, we conducted the following test scenario. We shut down 10 links from the network under test, and then measured the utilization of different links by applying a traffic matrix given in [24]. Fig. 5 compares the distribution of the load on different links of the network in our proposed traffic engineering method with that of CSPF. The x-axis shows the link utilization, and the y-axis shows the percentage of the links with the specified link utilization range.

It can be seen the number of links with more than 50% utilization in our method is much smaller than CSFP. We have used TOTEM toolbox [26] to find the test results for MIRA and CSPF.

Our proposed system is self-optimizing in the sense that its control loop is designed to engineer the traffic in such a way to minimize the network criticality. If a failure happens, the proposed control mechanism tries to locate the links with less utilization and assigns the demands to these links, so as to optimize the robustness (this shows the self-healing mechanism of our method as well). Further, if the fast loop of the system cannot find appropriate resources to assign to the demand, it will notify the long-term part of the management process, and appropriate resources will be assigned to the network accordingly. This provides for the self-configuring property of the system.

VII. Conclusion

In this paper we proposed an architecture to engineer the traffic of the network in a self-managed way. We used the concept of autonomic computing to build a two-loop control system which is capable of self-organizing, self-configuring, and self-healing. We have also analyzed the robustness of a network to the unexpected changes in different parameters and proposed an approach for path setup and routing of flows in our proposed system.

The essence of our work is based on determining a criticality index for each link/path showing how critical that link/path is to the changes in the topology and traffic demand of a network. We gave an analytical expression for the link and node criticality, and then proposed a heuristic for flow assignment based on it. Our algorithm identifies the least critical paths for allocation of new traffic flow requests. The results from applying the proposed algorithm to networks that are difficult to handle by existing approaches are very encouraging.

There are many issues that remain to be investigated in the new approach. We need to analyze the optimization problem of theorem 4.10 in detail and calculate the solution for criticality to the extent possible. This may give directions to provide a decentralized algorithm for flow-assignment. As another research challenge we need to look into the back up paths and the efficient algorithms to find them again with the goal of having less critical paths and back up paths. Finally, the SLA to weight mapping which is the main function of SLA interpreter is a complex task which needs to be carefully investigated.

Appendix A
Proof of Lemma 4.2

We use small English letters to show column vectors and small Greek letters to show row vectors. We also use subscript to show the order of a vector. For example z_{n-1} is a $n-1\times 1$ column vector and υ_{n-1} is a $1\times n-1$ row vector.

Without loss of generality, we rename the nodes so that the removed node becomes the last node of the graph (node n). Now, in order to write $L^{-1}(n|n)$ in terms of L, we use the Moore-Penrose generalized inverse matrix of L ([18]). The Moore-Penrose inverse of $L(n|n)$ and the $L^{-1}(n|n)$ are equal since $L(n|n)$ is an $(n-1)\times(n-1)$ matrix with rank $n-1$. In other words, $L(n|n)$ is full-rank and its inverse is the same as its Moore-Penrose inverse. To obtain L from $L(n|n)$, we first add a column to $L(n|n)$ to get $Q=[L(n|n)\ z_{n-1}]$. The column-vector z_{n-1} has to be chosen in a way to make the sum of every row of the matrix Q equal to zero. We use the following formula from [18] which is a recursive formula to obtain the Moore-Penrose inverse of a matrix when a column is added to the original matrix. Let $A\in\mathbb{F}^{p\times q}$ be a $p\times q$ matrix and $b\in\mathbb{F}^{p}$ be a $p\times 1$ column vector.

$$\begin{aligned}\begin{pmatrix}A & b_p\end{pmatrix}^{+} &= \begin{pmatrix}A^{+}(I-b_p\zeta_p)\\ \zeta_p\end{pmatrix} \\ \zeta_p &= \begin{cases}\frac{(b_p-AA^{+}b_p)^{+}}{b_p^{*}(AA^{*})^{+}} & if\ b_p\neq AA^{+}b_p\\ \frac{b_p^{*}(AA^{*})^{+}}{1+b_p^{*}(AA^{*})^{+}b_p} & b_p=AA^{+}b_p\end{cases}\end{aligned} \tag{16}$$

where $*$ means *conjugate transpose*. To satisfy the requirement of Laplacian matrix we need to have

$$[L(n|n)\ z_{n-1}]\overrightarrow{1}_{n-1}=0 \tag{17}$$

where $\overrightarrow{1}_{n-1}$ is a $(n-1)\times 1$ vector of all ones: $\overrightarrow{1}_{n-1}=[1\ 1\ 1\ ...\ 1]^{t}$. From (17) one can easily see that:

$$L(n|n)\overrightarrow{1}_{n-1}+z_{n-1}=0\Rightarrow z_{n-1}=-L(n|n)\overrightarrow{1}_{n-1} \tag{18}$$

Now from (16) by replacing $A=L(n|n)$ and using (18), one can see:

$$Q^{+}=\begin{pmatrix}L(n|n) & z_{n-1}\end{pmatrix}^{+}=\begin{pmatrix}L(n|n)^{+}\\ 0\end{pmatrix}+\overrightarrow{1}_{n-1}\zeta_{n-1}$$

It immediately follows that:

$$\Rightarrow (L^+(n|n))_{sk} = q^+_{sk} - q^+_{nk} \quad (19)$$

With the same approach , we add the n^{th} row to Q to obtain the $n \times n$ Laplacian matrix L: L $= \begin{bmatrix} Q \\ d \end{bmatrix}$ With similar reasoning and using equation (16) one can obtain:

$$\Rightarrow q^+_{sk} = l^+_{sk} - l^+_{sn} \quad (20)$$

Using equations (19), (20) we have: $(L^+(n|n))_{sk} = (L^{-1}(n|n))_{sk} = l^+_{sk} - l^+_{sn} - l^+_{nk} + l^+_{nn}$. A more detailed derivation can be found in [27].

APPENDIX B
PROOF OF COROLLARY 4.5

The average time that a random-walk starting at node s is in the system before reaches to its destination node d is equal to $T_{sd} = \sum_k b_{sk}(d)$. Now, the average time in system considering all possible source-destination pairs would be

$$\begin{aligned} T &= \frac{1}{n(n-1)} \sum_{s,d} T_{sd} = \frac{1}{n(n-1)} \sum_{s,d} \sum_k b_{sk}(d) \\ &= \frac{1}{n(n-1)} \sum_k \sum_{s,d} b_{sk}(d) = \frac{1}{n(n-1)} \sum_k b_k = \frac{B}{n-1} \end{aligned}$$

APPENDIX C
PROOF OF THEOREM 4.10

In order to proceed we need the following fact:

Lemma C.1: For any weight matrix W : $\nabla\tau.Vec(W) + \tau = 0$, where $Vec(W)$ is a vector obtained by concatenating all the rows of matrix W to get a vector of w_{ij}'s..

Proof: In lemma 4.3 we scale all the link weights with t

$$\tau(tVec(W)) = \frac{b(l)}{w(l)} = \frac{b(l)}{tw(l)} = \frac{1}{t}\tau(Vec(W)) \quad (21)$$

By taking the derivative of τ with respect to t, we will have

$$\nabla\tau(tVec(W)).Vec(W) = \frac{-1}{t^2}\tau(Vec(W)) \quad (22)$$

It is enough to consider equation (22) at $t = 1$ to get $\nabla\tau.Vec(W) + \tau = 0$.

■

In general, one can apply the condition of optimality [28], [29] on optimization problem (11) to get necessary condition for a weight vector to be optimal. Let W^* be the optimal weight matrix, and let W_t be another weight matrix satisfying the constraints of optimization problem (11), then according to the condition of optimality: $\nabla\tau.(Vec(W_t) - Vec(W^*)) \geq 0$. Now, we choose W_t as follows:

$$W_t = [w_{uv}] = \begin{cases} \frac{C}{2} & if \quad u = i \ \& \ v = j \\ \frac{C}{2} & if \quad u = j \ \& \ v = i \\ 0 & otherwise \end{cases}$$

Clearly, W_t satisfies the constraints of optimization problem (11), therefore, using the condition of optimality and considering lemma C.1 we have:

$$\begin{aligned} \nabla\tau.(Vec(W_t) - Vec(W^*)) &\geq 0 \\ \nabla\tau.Vec(W_t) - \nabla\tau.Vec(W^*) &\geq 0 \\ C\frac{\partial\tau}{\partial w_{ij}} + \tau &\geq 0 \quad \forall (i,j) \in E \quad (23) \end{aligned}$$

The constraints of optimization problem (11) and inequality (23) state necessary and sufficient conditions for the optimality of any weight matrix.

APPENDIX D
PROOF OF LEMMA 4.11

Since $\tau = \frac{b_{ij}}{w_{ij}}$, we have for $w_{ij} > 0$:

$$\frac{\partial\tau}{\partial w_{ij}} = \frac{1}{w_{ij}}\frac{\partial b_{ij}}{\partial w_{ij}} - \frac{\tau}{w_{ij}} \quad or \quad w_{ij}\frac{\partial\tau}{\partial w_{ij}} = \frac{\partial b_{ij}}{\partial w_{ij}} - \tau \quad (24)$$

By adding the results of equation (24) for different links of the network one can see:

$$\sum_{(i,j)\in E} w_{ij}\frac{\partial\tau}{\partial w_{ij}} = \sum_{(i,j)\in E}\frac{\partial b_{ij}}{\partial w_{ij}} - m\tau \quad (25)$$

Now it is enough to combine equation (25) and lemma C.1.

REFERENCES

[1] K. Kar, M. Kodialam, and T. V. Lakshman. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *IEEE Journal on Selected Areas in Communications*, 18(12):2566–2579, Dec. 2000.

[2] D. Applegate and E. Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: understanding fundamental tradeoffs. In *SIGCOMM*, pages 313–324, 2003.

[3] B. Fortz and M. Thorup. Increasing Internet Capacity Using Local Search. *Computational Optimization and Applications*, 29(12):13–48, 2004.

[4] K. Kar and T.V. Lakshman M. Kodialam. Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregated Link Usage Information. In *Proceedings of IEEE Infocom*, June 2001.

[5] R. Boutaba, W. Szeto, and Y. Iraqi. DORA: Efficient Routing for MPLS Traffic Engineering. *Journal of Network and Systems Management*, 10(3):309–325, September 2002.

[6] A. H. Dekker and B. D. Colbert. Network Robustness and Graph Topology. *Australasian Computer Science Conference*, 26:359–368, Jan. 2004.

[7] A. H. Dekker and B. Colberet. The Symmetry Ratio of a Network. In *Australasian symposium on Theory of computing*, volume 41, pages 13–20, Newcastle, Australia, 2005. ACM International Conference Proceeding Series.

[8] A. Tizghadam and A. Leon-Garcia. A Robust Routing Plan to Optimize Throughput in Core Networks. *ITC20, Elsvier*, pages 117–128, 2007.

[9] L. C. Freeman. Centrality in Networks: I. Conceptual Clarification. *Social Networks*, (1):215–239, 1978/79.

[10] M. Newman. A Measure of Betweenness Centrality Based on Random Walks. *arXiv cond-mat/0309045.*, 2003.

[11] C. Darwin. *The Origin of Species by Means of Natural Selection*. D. Appleton and Company, Available online at http://www.literature.org/authors/darwin-charles/the-origin-of-species/, 1859.

[12] A. Tizghadam and A. Leon-Garcia. Survival Value of Communication Networks. *Infocom Workshop on Network Science for Communications (NetSciCom)*, April 2009.

[13] P. Van Mieghem and F. A. Kuipers. Concepts of Exact QoS Routing Algorithms. *IEEE/ACM Transactions on Networking*, 12(5):851–864, October 2004.

[14] IBM. An Architectural Blueprint For Autonomic Computing, April 2003.

[15] Michael William Newman. *The Laplacian Spectrum of Graphs*. PhD thesis, Department of Mathematics, University of Manitoba, July 2000.

[16] Fan R. K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series On Mathematics, No. 92. American Mathematical society, 1997.

[17] D. J. Klein and M. Randic. Resistance Distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.

[18] Dennis S. Bernstein. *Matrix Mathematics*. Prinston University Press, 2005.

[19] M. Fiedler. Algebraic Connectivity of Graphs. *Czechoslovak Math. Journal*, 23(98):298–305, 1973.

[20] A. Bigdeli, A. Tizghadam, and A Leon-Garcia. Comparison of Network Criticality, Algebraic Connectivity, and other Graph Metrics. Venize, Italy, July 2009. First Workshop on Simplifying Complex Networks: SIMPLEX.

[21] L. Kleinrock. *Communication Nets, Stochastic Message Flow and Delay*. McGraw-Hill, New York, 1964.

[22] L. Kleinrock. *Queueing Systems*, volume II. John Wiley & Sons, 1975.

[23] B. Meister, H. R. Muller, and H. R. Rudin. New optimization criteria for message switching networks. *IEEE Transactions on Communication Technology*, 19(3):256–260, June 1971.

[24] http://totem.run.montefiore.ulg.ac.be/files/data/traffic-matrices-anonymized-v2.tar.bz2.

[25] M. Ziegelmann. *Constrained Shortest Paths and Related Problems - Constrained Network Optimization*. VDM Verlag, 2007.

[26] G. Leduca, H. Abrahamssone, S. Balona, S. Besslerb, M. D'Arienzoh, O. Delcourta, J. Domingo-Pascuald, S. Cerav-Erbasg, I. Gojmeracb, X. Masipd, A. Pescaph, B. Quoitinf, S.P. Romanoh, E. Salvadoric, F. Skivea, H.T. Tranb, S. Uhligf, and H. Umitg. An Open Source Traffic Engineering Toolbox. *Computer Communications*, 29(5):593–610, March 2006.

[27] A. Tizghadam and A. Leon-Garcia. On Robust Traffic Engineering in Core Networks. In *IEEE GLOBECOM*, December 2008.

[28] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, April 2003.

[29] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.