On Robust Network Planning

Ali Tizghadam
School of Electrical and Computer Engineering
University of Toronto, Toronto, Canada
Email: ali.tizghadam@utoronto.ca

Alberto Leon-Garcia
School of Electrical and Computer Engineering
University of Toronto, Toronto, Canada
Email: alberto.leongarcia@utoronto.ca

***Abstract*—One of the important properties of a reliable communication network is the robustness to the environmental changes. This paper looks at the design of robust networks from a new perspective. A graph-theoretical metric, betweenness, in combination with network weight matrix is used to define a global quantity, network criticality, to characterize the robustness of a network. We show that network criticality is a monotone decreasing function of weight matrix. Furthermore, it is shown that network criticality is a strictly convex function of network weight matrix. This leads to a well-defined convex optimization problem to find the optimal weight matrix assignment to minimize network criticality.**

I. Introduction

Robustness to the environmental changes is an important factor in the design of reliable communication networks. Robustness is the capability of a network to keep itself in a stable mode when environmental changes take place. Let us begin with our definition of robustness. There are three major types of environmental changes that may affect the performance of the network:

1) Changes in network topology including capacity.
2) Changes in community of interest, the set of active sources and sinks for traffic.
3) Changes in traffic demand.

Throughout this paper, we say that a "network" is robust if its performance is not sensitive to changes in topology, traffic or community of interest. In this sense, a robust network is more reliable, since unanticipated environmental changes (such as traffic shifts) cannot significantly impact the behavior of network. We aim to develop methods to design such robust networks with the help of graph-theoretical concepts.

In this work our approach is to find a metric to capture the effect of topology, traffic demand, and community of interest, and then design a network to control the sensitivity of the network to the shifts in traffic, or changes in topology by controlling the proposed metric. Using this approach we are able to investigate the problem analytically with the help of metrics from graph-theory. We can also discover some useful aspects of the robustness problem in networks.

The paper is organized as follows. Section II reviews previous work on network design and robustness in networks. In section III a review of our main metric, network criticality, is provided. The proposed optimization problem and its attributes are discussed in section IV, followed by some examples of network planning for well-defined graphs in section V. The paper is concluded in section VI.

II. Previous work

A wealth of literature is available on network robustness and its different aspects. [1] investigates the relationship between node similarity and optimal connectivity, and arrives at the result that a network provides maximum resistance to node destruction if it is both node-similar and optimally connected. The paper then describes a number of ways to design robust networks satisfying these conditions. But this paper considers only the effect of topology in the robustness of a network.

In [2] a way to design backbone networks is proposed that is insensitive to the traffic matrix (i.e., that works equally well for all valid traffic matrices), and that continues to provide guaranteed performance under a user-defined number of link and router failures. Valiant Load-Balancing method is used and the authors argue that it is a promising way to design robust backbone networks. The approach was first proposed by Valiant for processor interconnection networks [3], and has received recent interest for scalable routers with performance guarantees [4]. [2] applies Valiant method to backbone network design problem and provides appropriate capacity allocation for the links of a logical full mesh topology to support load-balancing for all possible traffic matrices. In the Valiant load-balancing method, traffic destined for a sink d is forwarded to intermediate hops with equal splits to all nodes, and then it is forwarded to the destination d. Delay propagation is one of the shortcomings of this method.

In [5] we proposed a framework for robust routing in core networks based on the idea of "link criticality" and "path criticality". Further development of the idea of criticality is studied in [6], where a mathematical framework for the definition of criticality is given within the context of Markov chain theory.

In this paper we quantify the robustness using the concept of criticality from [6]. We study the behavior of criticality and derive some useful results for the design of robust networks in the face of both topology and traffic variations.

III. Network Criticality

In this section we summarize the results of our previous work on robustness [6].

A. Network Model

We model a network with an undirected weighted graph $G = (N, E, W)$ where N is the set of nodes, E is the set of graph links, and W is the weight matrix of the graph. Throughout this paper we assume that G is a connected graph. We assume that SLA (Service Level Agreement) parameters are already mapped to the weights by some appropriate method. Some of these methods are discussed in [7]. This permits us to abstract different business policies and/or SLA's as parts of the weight definition. In this paper we are interested in the study of the weights and their effect on robustness.

In [8] a probabilistic interpretation of the betweenness is defined based on random walks. The betweenness 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. The random-walk is defined on a Markov chain M with transition probability matrix P according to the following rule:

$$p_{ij} = \begin{cases} \frac{w_{ij}}{\sum_{k\in A(i)} w_{ik}} & if \quad j \in A(i) \\ 0 & otherwise \end{cases} \tag{1}$$

where $A(i)$ is the set of adjacent nodes of i and $w_{ik} \geq 0$ is the weight of link (i,k).

B. Definition of Network Criticality

We now introduce *network criticality*, the metric that we proposed in [6], to quantify the robustness of a network. We start by defining node/link criticality.

Definition 3.1: Node criticality is defined as the random-walk betweenness of a node over its weight (weight of a node is defined as the sum of the weights of its incident links). Link criticality is also defined as the betweenness of a link over its weight.

Let η_k be the criticality of node k and η_{ij} be the criticality of link $l=(i,j)$. It is shown in [6] that η_i and η_{ij} can be obtained by the following expressions:

$$\begin{aligned} \tau_{sd} &= l^+_{ss} + l^+_{dd} - 2l^+_{sd} \quad or \quad \tau_{sd} = u^t_{sd} L^+ u_{sd} && (2)\\ \tau &= \sum_s \sum_d \tau_{sd}, \quad \bar{\tau} = \frac{1}{n(n-1)}\tau && (3)\\ \eta_k &= \frac{b_k}{W_k} = \frac{1}{2}\tau = \frac{n(n-1)}{2}\bar{\tau} && (4)\\ \eta_{ij} &= \frac{b_{ij}}{w_{ij}} = \tau = n(n-1)\bar{\tau} && (5)\end{aligned}$$

where L^+ is the Moore-Penrose inverse of graph Laplacian matrix L [9], n is the number of nodes, and $u_{ij} = [0 \ldots 1 \ldots -1 \ldots 0]^t$ (1 and -1 are in i^{th} and j^{th} position).

Observation 3.2: Equations (2) to (5) show that node criticality (η_k) and link criticality (η_{ij}) are independent of the node/link position and only depend on τ (or $\bar{\tau}$) which is a global quantity of the network.

Definition 3.3: We refer to $\bar{\tau}$ as the normalized network criticality or simply *network criticality*.
One can see that $\bar{\tau}$ is a global quantity on network graph G. Equations (4) and (5) show that node (link) betweenness consists of a local parameter (weight) and a global metric (network criticality). $\bar{\tau}$ can capture the effect of topology and community of interest via betweenness, and the effect of traffic via weight (by appropriate definition of weight). The higher the betweenness of a node/link, the higher the risk of using the node/link. Furthermore, one can define node/link capacity as the weight of a node/link, then the higher the weight of a node/link, the lower the risk of using the node/link. Therefore network criticality can quantify the risk of using a node/link in a network which in turn indicates the degree of robustness of the network.

This motivates the rest of our work in this paper. We investigate network criticality as a network-wide metric to capture and optimize network robustness. In this paper our goal is to investigate $\bar{\tau}$ as a function of weight matrix (W) and find necessary conditions for its optimality. We aim to find an appropriate weight matrix that can optimize the robustness of a network.

IV. Optimization of Network Criticality

In this section we investigate the behavior of $\bar{\tau}$ as a function of the weight matrix.

A. Monotonicity and Convexity of $\bar{\tau}$

We start by investigating the convexity of $\bar{\tau}$. In order to proceed we need the following lemma.

Lemma 4.1: Network criticality ($\bar{\tau}$) is a monotone decreasing function of weights. More precisely

$$\frac{\partial \bar{\tau}}{\partial w_{ij}} = -\frac{2}{n-1} \| L^+_i - L^+_j \|^2$$

Furthermore, the second partial derivative of $\bar{\tau}$ with respect to the weight w_{ij} is non-negative and can be found from the following equation:

$$\frac{\partial^2 \bar{\tau}}{\partial {w_{ij}}^2} = -2\tau_{ij} \frac{\partial \bar{\tau}}{\partial w_{ij}}$$

Proof: See Appendix A. ▮

■

Theorem 4.2: Network criticality ($\bar{\tau}$) is a strictly convex function of weights.

Proof: It's enough to show that the first derivative of $\bar{\tau}$ is always negative and it's second derivative is always non-negative. Considering the result of lemma 4.1, these conditions are met in $\bar{\tau}$. ▮

■

B. Optimization of $\bar{\tau}$

Since $\bar{\tau}$ is a strictly convex function of the weights, an optimization problem with linear constraints has a unique solution.

We consider the minimization of τ under some constraints. We set the following as a constraint for our optimization problem: $\sum_{(i,j)\in E} z_{ij} w_{ij} \leq C$, where z_{ij} is the cost of link (i,j). One can consider C as the allowable budget for total network weight. Consider the following optimization problem on graph $G(N,E,W)$:

$$\begin{array}{lll} & Minimize & \bar{\tau} \\ Subject\ to & \sum_{(i,j)\in E} z_{ij} w_{ij} \leq C & ,C\ is\ fixed \\ & w_{ij} \geq 0 & \end{array} \tag{6}$$

Theorem 4.3: For the optimal weight set, W^*, in optimization problem (6) we have:

$$w^*_{ij}(C\frac{\partial\bar{\tau}}{\partial w_{ij}} + z_{ij}\bar{\tau}) = 0 \quad \forall\ (i,j) \in E \tag{7}$$

Proof: We will prove theorem 4.3 using the following lemma.

Lemma 4.4: For any weight matrix W of links of a graph:

$$Vec(W)^t\nabla\bar{\tau} + \bar{\tau} = 0$$

where $Vec(W)$ is a vector obtained by concatenating all the rows of matrix W to get a vector of w_{ij}'s.

Proof: Suppose we scale all the link weights in a graph with factor t, then using equation (5) we have:

$$\bar{\tau}(tVec(W)) \quad = \quad \frac{1}{n(n-1)}\frac{b_{ij}}{tw_{ij}}$$

Note that the transition probabilities are invariant to the scaling of the weights based on their definition in equation (1). This implies the invariance of link betweenness b_{ij}. Therefore:

$$\bar{\tau}(tVec(W)) \quad = \quad \frac{1}{t}\bar{\tau}(Vec(W)) \tag{8}$$

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

$$Vec(W)^t\nabla\bar{\tau} = \frac{-1}{t^2}\bar{\tau}(Vec(W)) \tag{9}$$

It is enough to consider equation (9) at $t = 1$ to obtain $Vec(W)^t\nabla\bar{\tau} + \bar{\tau} = 0$. ▮

■

In optimization problem (6), $\bar{\tau}$ is a continuous decreasing function of link weights and it is strictly convex. This implies that in first constraint of problem (6) the inequality can be replaced by equality.

Now, to conclude the proof of theorem 4.3 we notice that for the optimal weight matrix W^* we can write

$$(Vec(Z).Vec(W^*))\bar{\tau} = (\sum_{(i,j)\in E} w^*_{ij}z_{ij})\bar{\tau} = C\bar{\tau} \tag{10}$$

Combining lemma 4.4 and equation (10) one can see

$$\begin{aligned}&C\nabla\bar{\tau}.Vec(W^*) + Vec(Z).Vec(W^*)\bar{\tau} = 0\\ &Vec(W^*).(C\nabla\bar{\tau} + \bar{\tau}Vec(Z)) = 0\\ &w^*_{ij}(C\frac{\partial\bar{\tau}}{\partial w_{ij}} + \bar{\tau}z_{ij}) = 0 \quad \forall\ (i,j) \in E\end{aligned}$$

This completes the proof of theorem 4.3. ▮

■

Equation (7) gives the necessary condition for the optimality of convex problem (6).

Optimization problem (6) can be converted to a semi-definite programming. In order to find the semi-definite programming representation of the optimization problem (6), we need the following lemma.

Lemma 4.5: $\bar{\tau} = \frac{2}{n-1}Tr(L^+)$.

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

$$\begin{aligned}\tau &= n\sum_i l^+_{ii} + n\sum_i l^+_{ii} - 2\times 0 = 2n\sum_i l^+_{ii} = 2nTr(L^+)\\ \bar{\tau} &= \frac{1}{n(n-1)}2nTr(L^+) = \frac{2}{n-1}Tr(L^+)\end{aligned}$$

This completes the proof of lemma 4.5. ▮

■

Suppose we let $\Gamma = (L+\frac{J}{n})^{-1}$, where J is a n-by-n matrix with all elements equal to 1, then Γ can be written as a semi-definite inequality as follows. 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 ([10]) be positive semi-definite. In general, the Schur complement of a matrix of the form $\begin{pmatrix}A & B\\ C & D\end{pmatrix}$ is: $A - BD^{-1}C$. Hence the Schur complement of Θ is $\Gamma - (L+\frac{J}{n})^{-1}$, and

$$\Theta = \begin{pmatrix}\Gamma & I\\ I & L+\frac{J}{n}\end{pmatrix} \geq 0 \Leftrightarrow \Gamma \geq (L+\frac{J}{n})^{-1} \tag{11}$$

where $\geq$ means positive semi-definite. Since the optimization problem (6) should minimize $Tr(\Gamma)$, the equality in equation (11) is chosen which is equivalent to $\Gamma = (L+\frac{J}{n})^{-1}$. Now optimization problem (6) can be converted to a semi-definite programming:

$$\begin{aligned}Minimize \quad & \frac{2}{n-1}Tr(\Gamma) - \frac{2}{n-1} \\ Subject\ to \quad & \sum_{i,j} z_{ij}w_{ij} = C\\ & \begin{pmatrix}\Gamma & I\\ I & L+\frac{J}{n}\end{pmatrix} \geq 0\end{aligned} \tag{12}$$

This new optimization problem can be solved with standard methods of solving semi-definite programs.

In this paper we solve this optimization problem for some specific graphs to show how the concept of network criticality helps find robust network topologies.

C. Capacity Planning

In this section we consider the capacity planning as an important special case of network design problem. Consider a network $G(N,E,W)$ where the link weights are equal to the link capacities, 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 traffic load $\gamma_{ij}\ \forall(i,j) \in E$ 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. Let z_{ij} be the symmetric cost of assigning capacity c_{ij} to link (i,j), and suppose that we have a linear cost function. The total cost of the capacity assignment problem is $\sum_{(i,j)\in E} z_{ij}c_{ij}$. We fix this total cost to C. We can write the optimization problem for capacity assignment problem as follows:

$$\begin{aligned}& Minimize \quad \bar{\tau}\\ Subject\ to \quad & \textstyle\sum_{(i,j)\in E} c_{ij}z_{ij} = C \quad ,C\ is\ fixed\\ & c_{ij} \geq \gamma_{ij}\end{aligned} \tag{13}$$

By applying the change of variable $c_{ij} = c'_{ij} + \gamma_{ij}$ to the optimization problem (13), we will have the following convex optimization problem.

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

where $C' = C - \sum_{(i,j)\in E} z_{ij}\gamma_{ij}$. The optimization problem (14) is now converted to the optimization problem (6) (with $w_{ij} \to c'_{ij}$ and $C \to C'$), therefore, all the results of this section are applicable for the capacity assignment problem. Later in this chapter we will see when the optimization problem (14) is equal to the Kleinrock's capacity assignment problem [11].

V. Case study

In this section we investigate the optimal network weight allocation for some well-defined network topologies. In the following examples if we assume that the weight of a link is equal to its capacity, then the problem is converted to a capacity assignment problem.

A. Complete Graph on n Nodes (K_n)

For K_n we can obtain the solution of optimization problem (6) analytically. In this example we assume $z_{ij} = 1 \ \ \forall\ (i,j) \in E$. In order to find the optimal weight set for K_n we need the following lemmas.

Lemma 5.1: $\bar{\tau}$ can be written as: $\bar{\tau} = \frac{2}{n-2}\sum_{i=2}^{n}\frac{1}{\lambda_i}$, where $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are eigenvalues of graph Laplacian L.

Proof: We know from linear algebra that the sum of eigenvalues of a square matrix is equal to the trace of the matrix. On the other hand, the non-zero eigenvalues of L^+ are reciprocals of the non-zero eigenvalues of L.Consequenty, Lemma 5.1 will be a direct result of Lemma 4.5. ▮

■

Lemma 5.2: Consider optimization problem (12) and suppose all the links have equal costs, that is, $Z = J - I$ (this is a square-matrix whose entries are all 1, except the diagonal elements which are set to zero) . If there is an automorphism on a graph G(N,E,W) that can map link $l = (i,j)$ on link $l' = (i',j')$, then these links should have equal optimal weights.

Proof: Let G' be the new graph after applying the automorphism. Any automorphism on graph G can be shown by a matrix T so that the Laplacian of transformed graph G' can be obtained from Laplacian of original graph G as $L(G') = TL(G)T^t$. This means that L(G) and L(G') have the same eigenvalues. As a result according to the Lemma 5.1 criticality of graph G and G' are the same: $\bar{\tau}(G) = \bar{\tau}(G')$. On the other hand the solution of optimization problem (6) is unique. As a result the weight of link l and l' are the same. ▮

■

Corollary 5.3: Consider optimization problem (12) and assume that the graph of the network is an edge-transitive graph with equal link costs. The optimal weight for a link $(i,j) \in E$ is equal to $w_{ij} = \frac{C}{m}$, where m denotes the number of graph links.

Proof: A graph is edge-transitive, if there is an automorphism that can map any two links of the graph. According to lemma 5.2 all the link weights are equal. In addition, suppose $w_{ij} = w \ \forall\ (i,j) \in E$, then constraint $\sum_{(i,j)\in E} w_{ij} = C$ implies that $w = \frac{C}{m}$. This completes the proof of corollary 5.3. ▮

■

Complete graph K_n is an edge-transitive graph, therefore, according to corollary 5.3 the optimal weight of all the links of K_n are equal. Let denote this common weight by w. Further, let vector X be the eigenvector of Laplacian matrix for eigenvalue λ. Then:

$$L(K_n) = w(nI - J) \quad and \quad LX \ = \ \lambda X$$
$$w((n-1)x_j - \textstyle\sum_{i\neq j} x_i) = \lambda x_j \quad for \quad i \neq 1$$

In addition $\sum_{i=1}^{n} x_i = 0$ (property of the Laplacian matrix for K_n). Therefore

$$\begin{aligned} w((n-1)x_j - (-x_j)) &= \lambda x_j \\ \lambda_i &= nw \quad for \quad i = 2\ ...\ n \end{aligned}$$

One can also find link weight w from corollary 5.3. The total number of links in K_n is $m = n(n-1)$, therefore, according to the corollary 5.3 $w = \frac{1}{n(n-1)}C$. Therefore

$$\lambda_i = \frac{1}{n-1}C \quad \forall\ i \in N \qquad (15)$$

It is easy now to calculate network criticality for graph K_n using Lemma 5.1 and equation (15).

$$\bar{\tau} = \frac{2}{n-1}\sum_{i=2}^{n}\frac{1}{\lambda_i} = \frac{2(n-1)}{C} \qquad (16)$$

According to equation (16) the optimal network criticality in a complete graph is linearly increasing with the size of the network. This provides a basis for comparing the normalized network criticality of different networks against the full-mesh on n nodes.

1) Case of Unequal Link Costs for K_n: When the link costs (z_{ij}'s) are not equal, we can use the semi-definite approach to find the best weight assignment. We use a numerical example to show the effect of changes in link costs. We consider the complete graph on 6 nodes (K_6), and we assume that the matrix of link costs is given as follows:

$$Z = [z_{ij}] = \begin{pmatrix} 0 & 1.2 & 1 & 1 & 1 & 2 \\ 1.2 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 2 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Further, let $C = 2000$. We used the semi-definite form of the optimization problem which is described in equation (12) and solved semi-definite program for complete graph K_6 using CVX, a package for specifying and solving convex programs [12], [13]. The optimal weight assignment is given in the

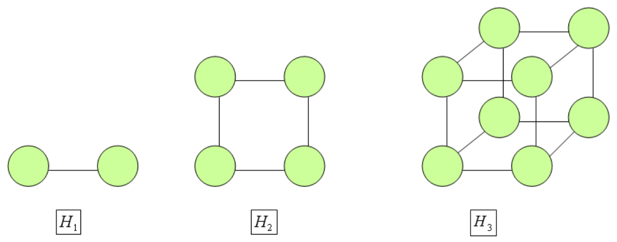

Fig. 1: Hypercube Topology (H_1, H_2, H_3)

following.

$$W = \begin{pmatrix} 0 & 54.982 & 85.765 & 85.746 & 85.738 & \mathbf{0.003} \\ 54.982 & 0 & 64.057 & 63.995 & 63.975 & 78.027 \\ 85.765 & 64.057 & 0 & 54.321 & 54.233 & 81.188 \\ 85.746 & 63.995 & 54.321 & 0 & 54.435 & 81.258 \\ 85.738 & 63.975 & 54.233 & 54.435 & 0 & 81.276 \\ \mathbf{0.003} & 78.027 & 81.188 & 81.258 & 81.276 & 0 \end{pmatrix}$$

The weight matrix shows that the optimal weight assignment is not uniform. The optimal weight of link $(1, 6)$ is $w_{16} = w_{61} = 0.003$ which means that link $(1, 6)$ is effectively down. In other words, the topology of the optimal network is not K_6 any more.

B. Hypercube with 2^n nodes (H_n)

Now we consider hypercube of order n (H_n), another well-structured graph whose criticality can be obtained analytically. In this example we assume $z_{ij} = 1 \quad \forall\ (i, j) \in E$. Hypercube is an edge-transitive graph, therefore, by corollary 5.3 the optimal solution of the optimization problem (12) for hypercube has equal weights. Hence, we consider a hypercube with weight w for all the links. Hypercube can be recursively built by the use of "Cartesian Product" of a graph with K_2 (complete graph on 2 nodes):

$$H_{n+1} = H_n \square K_2;$$

where $\square$ denotes the cartesian product. This equation can also be written using "Kronecker Product":

$$\begin{aligned} H_{n+1} &= H_n \square K_2; \\ &= H_n \otimes I_2 + I_{2^n} \otimes K_2 \qquad (17) \end{aligned}$$

We have used the symbol $\otimes$ to denote Kronecker product. Fig. 1 shows hypercube topology for $n = 1\ to\ 3$.

We try to obtain eigenvalues of adjacency matrix of H_n using equation (17). We find the eigenvalues for $w = 1$, then we multiply these normalized eigenvalues by w to find the eigenvalues for the general case.

$$\begin{aligned} H_{n+1} &= H_n \otimes I_2 + I_{2^n} \otimes K_2 \\ &= \begin{pmatrix} H_n & 0 \\ 0 & H_n \end{pmatrix} + \begin{pmatrix} 0 & I_{2^n} \\ I_{2^n} & 0 \end{pmatrix} \\ H_{n+1} &= \begin{pmatrix} H_n & I_{2^n} \\ I_{2^n} & H_n \end{pmatrix} \end{aligned}$$

For simplicity of notation, we drop the subscript from I_{2^n} and use I instead, which means the identity matrix of appropriate order. Now we try to build the determinant of characteristic matrix $H_{n+1} - \lambda I$:

$$\begin{aligned} H_{n+1} - \lambda I &= \begin{pmatrix} H_n - \lambda I & I \\ I & H_n - \lambda I \end{pmatrix} \\ d_{n+1} &= |H_{n+1} - \lambda I| = \det \begin{pmatrix} H_n - \lambda I & I \\ I & H_n - \lambda I \end{pmatrix} \\ &= \det \begin{pmatrix} I & H_n - \lambda I \\ H_n - \lambda I & I \end{pmatrix} \end{aligned}$$

Now we multiply the first row by $H_n - \lambda I$, and then subtract the first row from the second row. We have:

$$\begin{aligned} d_{n+1}(\lambda) &= \det \begin{pmatrix} I & H_n - \lambda I \\ H_n - \lambda I - (H_n - \lambda I) & I - (H_n - \lambda I)^2 \end{pmatrix} \\ &= \det \begin{pmatrix} I & H_n - \lambda I \\ 0 & I - (H_n - \lambda I)^2 \end{pmatrix} \\ &= |I - (H_n - \lambda I)^2| \\ &= |H_n - (\lambda - 1)I||H_n - (\lambda + 1)I| \\ d_{n+1}(\lambda) &= d_n(\lambda - 1)d_n(\lambda + 1) \end{aligned}$$

Using this recursive formula for determinant of hypercube, one can find with induction that the eigenvalues of H_n are $2k - n, \quad k = 0, 1, ..., n$ with multiplicity $C(n, k) = \frac{n!}{k!(n-k)!}$. This result is true when all the weights are set to 1. In general case where we have a weight w for each link, the eigenvalue is also multiplied by this weight.

We notice that hypercube is a regular graph (degree of all nodes are n). This means that we can find eigenvalues of Laplacian using eigenvalues of the adjacency matrix of H_n:

$$\begin{aligned} L_n &= nI - H_n \\ \lambda_k &= n - (2k - n)\ with\ multiplicity\ C(n, k) \\ \lambda_k &= 2(n - k)\ with\ multiplicity\ C(n, k) \end{aligned}$$

Now one can easily find the network criticality for H_n using lemma 5.1.

$$\begin{aligned} \bar{\tau} &= \frac{2}{2^n - 1} \sum_k \frac{1}{\lambda_k} \\ &= \frac{2}{2^n - 1} \sum_{k=0}^{n-1} \frac{C(n, k)}{2(n - k)w} \\ &= \frac{1}{(2^n - 1)w} \sum_{k=0}^{n-1} \frac{C(n, k)}{n - k} \qquad (18) \end{aligned}$$

On the other hand, by considering the fact that the number of links in H_n is $m = n2^n$, by corollary 5.3, we have:

$$w = \frac{C}{n2^n} \qquad (19)$$

The final expression for network criticality of H_n can be found

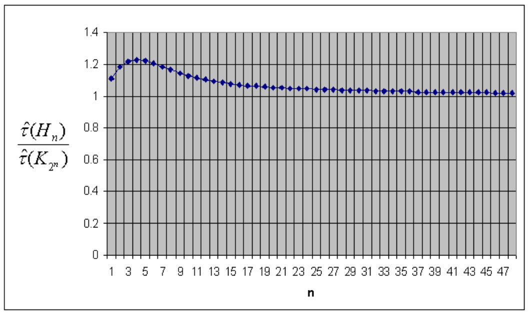

Fig. 2: The Ratio of Normalized Network Criticality of Hypercube and Complete Graph

by applying equation (19) in (18):

$$\bar{\tau} = \frac{n2^n}{(2^n-1)C}\sum_{k=0}^{n-1}\frac{C(n,k)}{n-k}$$

$$\bar{\tau} = \frac{n}{(1-\frac{1}{2^n})C}\sum_{i=1}^{n}\frac{C(n,i)}{i} \tag{20}$$

To obtain the last equation we applied the change of variable $i = n-k$ and used the fact that $C(n, n-i) = C(n, i)$. Equation (20) shows the behavior of normalized network criticality when the size of hypercube increases.

We can compare the normalized criticality of a hypercube H_n with a complete graph K_{2^n} to see how the robustness is decreased by changing a complete graph to a hypercube (with the same number of nodes).

$$\frac{\bar{\tau}(H_n)}{\bar{\tau}(K_{2^n})} = \frac{\frac{n}{(1-\frac{1}{2^n})C}\sum_{i=1}^{n}\frac{C(n,i)}{i}}{\frac{2(2^n-1)}{C}}$$

$$\rightarrow \frac{n}{2^{n+1}}\sum_{i=1}^{n}\frac{C(n,i)}{i} \tag{21}$$

Fig. 2 shows the graphical behavior of equation (21) for different values of n. It can be seen that for higher values of n, fraction $\frac{\bar{\tau}(H_n)}{\bar{\tau}(K_{2^n})}$ approaches 1. Note that even for high values of n the difference between the normalized criticality of H_n and K_{2^n} is considerable, although the ratio is decreasing.

C. Optimal Network Criticality for a Tree

We note that a tree is an acyclic simple graph, which means that there is exactly one path between every two nodes of a tree. It follows that network criticality of a tree can be found from the following equation.

$$\tau = \sum_{(i,j)\in E}\frac{n_{ij}}{w_{ij}} \tag{22}$$

$$\frac{\partial\tau}{\partial w_{ij}} = -\frac{n_{ij}}{w_{ij}^2} \tag{23}$$

where n_{ij} denotes the number of times that link (i, j) is in the path connecting any source to any destination. But from equation (7) we know that for optimal weights $C\frac{\partial\tau}{\partial w_{ij}} + \tau z_{ij} = 0$. By using this relation in equation (23) we get:

$$\frac{\partial\tau}{\partial w_{ij}} = -\frac{n_{ij}}{w_{ij}^2} = -\frac{z_{ij}\tau}{C} \tag{24}$$

$$\Rightarrow w_{ij} = (\frac{n_{ij}C}{z_{ij}\tau})^{\frac{1}{2}} \tag{25}$$

From the constraint of the optimization problem we have $\sum_{(i,j)\in E} z_{ij}w_{ij} = C$. Hence:

$$\sum_{(i,j)\in E}(\frac{n_{ij}z_{ij}C}{\tau})^{\frac{1}{2}} = C \tag{26}$$

$$\tau = (\sum_{(i,j)\in E}(\frac{n_{ij}z_{ij}}{C})^{\frac{1}{2}})^2 \tag{27}$$

Now it is enough to substitute τ from equation (27) in equation (25) to have optimal weight for tree.

$$w_{ij} = (\frac{n_{ij}C}{z_{ij}})^{\frac{1}{2}} \times \frac{1}{\sum_{(i,j)\in E}(\frac{n_{ij}z_{ij}}{C})^{\frac{1}{2}}}$$

Finally

$$w_{ij} = \frac{C}{z_{ij}} \times \frac{(n_{ij}z_{ij})^{\frac{1}{2}}}{\sum_{(i,j)\in E}(n_{ij}z_{ij})^{\frac{1}{2}}} \tag{28}$$

Equation (28) shows that the optimal weight of a link in a tree is proportional to the square root of n_{ij}.

1) Capacity Planning for a Tree: The capacity assignment problem for a tree can be solved analytically using the guidelines provided in section IV-C. It is enough to apply the following changes in equation (28):

$$w_{ij} \rightarrow c_{ij} - \gamma_{ij}$$

$$C \rightarrow C - \sum_{(i,j)\in E} z_{ij}\gamma_{ij}$$

The optimal capacity assignment for a tree would be:

$$c_{ij} = \gamma_{ij} + \frac{C - \sum_{(i,j)\in E} z_{ij}\gamma_{ij}}{z_{ij}} \times \frac{(n_{ij}z_{ij})^{\frac{1}{2}}}{\sum_{(i,j)\in E}(n_{ij}z_{ij})^{\frac{1}{2}}} \tag{29}$$

There is a close analogy between our result and Kleinrock's result for capacity assignment. In [11] Kleinrock showed that under the independence assumption the optimal capacity (to minimize average delay of the network) of a link is proportional to the square root of the link rate. Note that for a tree n_{ij} is proportional to the link load (γ_{ij}) since there is only one path between every source-destination pair. As a result, equation (29) is similar to the Kleinrock's equation for optimal capacity ([11], §5.7, equation 5.26). This result is not surprising because the network criticality of a tree according to equation (22) is equal to $\tau = \sum_{(i,j)\in E}\frac{n_{ij}}{c_{ij}-\gamma_{ij}}$ (considering $w_{ij} = c'_{ij}$). This is the same expression that is used in [11] to find the average delay of a network ([11], §5.6, equation 5.19), therefore, the minimization of network criticality is equal to the minimization of the average network delay when the network is a tree.

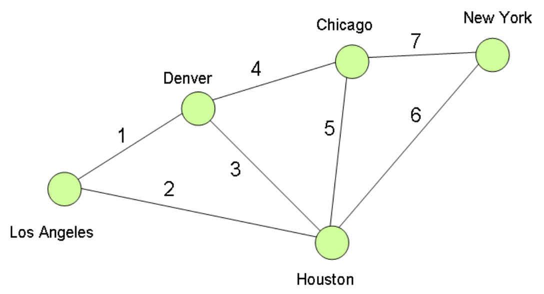

Fig. 3: Kleinrock's Network

D. *Kleinrock's Network*

In this following the proposed optimal weight assignment method is compared with Kleinrock's method for capacity assignment [14], [11] and Meister's extension [15] using the example of telegraph network in Kleinrock illustrated in Fig. 3(see [14], pp. 22-23). In this example the link cost factors z_{ij} are all considered equal to unity. 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 which 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 it 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 and Meister), 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). Table III shows individual link values for three methods. Kleinrock's method assigns very large delay to link 3 because the demand on link 3 is much less than other links. Meister's method assigns equal delays for all the links. This resolves the issue with Kleinrock's method, but introduces a fairness problem. In our proposed method, the link delays are nt 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.

Link	Load	Kleinrock	Meister	Criticality Method
1	3.15	27.93	27.00	29.63
2	3.55	29.85	27.40	33.31
3	0.13	5.16	23.98	12.67
4	3.64	30.28	27.49	32.95
5	0.82	13.46	24.67	13.36
6	3.88	31.38	27.73	33.64
7	9.95	53.99	33.80	36.43

TABLE I: Capacity Assignment using 3 Different Methods

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

TABLE II: Average Network Delay and Network Criticality using Different Methods

Link	Kleinrock	Meister	Criticality Method
1	40.36	41.93	37.76
2	38.02	41.93	33.60
3	198.67	41.93	79.71
4	37.54	41.93	34.12
5	79.10	41.93	79.71
6	36.36	41.93	33.60
7	22.71	41.93	37.76

TABLE III: Individual Link Delays using 3 Different Methods

VI. Conclusion

In this paper we proposed an approach to the network design problem and network planning using graph-theoretical concepts. We used network criticality metric to quantify the robustness of a network and investigated the properties of network criticality. We showed that network criticality is a strictly convex function of link weights and investigated the convex optimization problem of minimizing the network criticality under some constraints on the weight matrix. We also found a semi-definite programming representation of this problem which permits us to use available literature on semi-definite programming to solve the optimization problem and find the optimal weights. Capacity assignment problem can be considered as a special case of this general problem where the weight of a link is equal to its capacity.

Appendix A
Proof of Lemma 4.1

We can calculate the first derivative of τ_{sd} with respect to the weight of a typical link $l = (i, j)$ using equation (2):

$$\frac{\partial \tau_{sd}}{\partial w_{ij}} = u_{sd}^{t} \times \frac{\partial L^{+}}{\partial w_{ij}} \times u_{sd} \qquad (30)$$

Now, we use the following fact from graph-theory about generalized inverse Laplacian of a graph [16].

$$L^{+} = (L + \frac{J}{n})^{-1} - \frac{J}{n} \qquad (31)$$

Matrix J in this equation is an $n \times n$ matrix with all elements equal to 1. Using equation (31) we have: $\frac{\partial L^+}{\partial w_{ij}} = \frac{\partial (L+\frac{J}{n})^{-1}}{\partial w_{ij}}$. On the other hand:

$$
\begin{aligned}
&(L + \frac{J}{n})^{-1} \times (L + \frac{J}{n}) = I \\
&\frac{\partial (L + \frac{J}{n})^{-1}}{\partial w_{ij}} \times (L + \frac{J}{n}) + (L + \frac{J}{n})^{-1} \times \frac{\partial (L + \frac{J}{n})}{\partial w_{ij}} = 0 \\
&\frac{\partial L^+}{\partial w_{ij}} = -(L + \frac{J}{n})^{-1} \times \frac{\partial L}{\partial w_{ij}} \times (L + \frac{J}{n})^{-1} \qquad (32)
\end{aligned}
$$

Replacing equation (32) in equation (30) will result in:

$$
\begin{aligned}
\frac{\partial \tau_{sd}}{\partial w_{ij}} &= u_{sd}^t \times \frac{\partial L+}{\partial w_{ij}} \times u_{sd} \\
&= -1 \times u_{sd}^t \times (L + \frac{J}{n})^{-1} \times \frac{\partial L}{\partial w_{ij}} \\
&\quad \times (L + \frac{J}{n})^{-1} \times u_{sd} \qquad (33)
\end{aligned}
$$

To obtain $\frac{\partial L}{\partial w_{ij}}$ we notice that in four elements of matrix L the weight w_{ij} appears: $l_{ij}, \ , l_{ji}, \ , l_{ii}, \ , l_{jj}, \ .$ Based on the definition of Laplacian ($L = \sum_{(i,j)\in E} w_{ij} u_{ij} u_{ij}^t$ [9]) we have:

$$
\frac{\partial L}{\partial w_{ij}} = u_{ij} \times u_{ij}^t \qquad (34)
$$

Combining equations (34) and (33) we have

$$
\begin{aligned}
\frac{\partial \tau_{sd}}{\partial w_{ij}} &= -1 \times u_{sd}^t \times (L + \frac{J}{n})^{-1} \\
&\quad \times u_{ij} \times u_{ij}^t \times (L + \frac{J}{n})^{-1} \times u_{sd} \qquad (35)
\end{aligned}
$$

One can also easily verify that:

$$
(L + \frac{J}{n})^{-1} u_{ij} = ((L + \frac{J}{n})^{-1} - \frac{J}{n}) u_{ij} = L^+ u_{ij} = L_i^+ - L_j^+ \quad (36)
$$

where L_i^+ is the i^{th} column of L^+. Here we used the fact that $J u_{ij} = u_{ij}^t J = 0$. Using equation (36) in (35) gives:

$$
\begin{aligned}
\frac{\partial \tau_{sd}}{\partial w_{ij}} &= -1 \times u_{sd}^t (L_i^+ - L_j^+)(L_i^+ - L_j^+)^t u_{sd} \\
&= -1 \times ((L_i^+ - L_j^+)^t u_{sd})^t (L_i^+ - L_j^+)^t u_{sd} \\
&= -1 \times ((l_{is}^+ - l_{id}^+) - (l_{js}^+ - l_{jd}^+))^2 \\
&= -1 \times ((l_{is}^+ - l_{js}^+)^2 + (l_{id}^+ - l_{jd}^+)^2 \\
&- \quad 2(l_{is}^+ - l_{js}^+)(l_{id}^+ - l_{jd}^+))
\end{aligned}
$$

Now we are ready to obtain the derivative of τ.

$$
\begin{aligned}
\frac{\partial \tau}{\partial w_{ij}} &= \sum_d \sum_s \tau_{sd} \\
&= -1 \times (\sum_d \sum_s (l_{is}^+ - l_{js}^+)^2 + \quad \sum_d \sum_s (l_{id}^+ - l_{jd}^+)^2 - \\
&\quad 2 \sum_d \sum_s (l_{is}^+ - l_{js}^+)(l_{id}^+ - l_{jd}^+)) \\
&= -1 \times (n \|L^+ u_{ij}\|^2 + n \|L^+ u_{ij}\|^2 - \\
&\quad 2 \sum_d (l_{id}^+ - l_{jd}^+) \sum_s (l_{is}^+ - l_{js}^+)) \\
&= -2n \|L^+ u_{ij}\|^2 \\
\frac{\partial \tau}{\partial w_{ij}} &= -2n \|L_i^+ - L_j^+\|^2 \qquad (37)
\end{aligned}
$$

Equation (37) shows that the derivative of $\bar{\tau}$ is always non-positive. It is also non-zero since if it would be zero for some weights, it would mean that two columns of L^+ are equal. Another way to put this is to say that the rank of L^+ is n-2 while the rank of L^+ has to be n-1 in order to guarantee connectivity of the graph.

$$
\begin{aligned}
\frac{\partial^2 \bar{\tau}}{\partial {w_{ij}}^2} &= -\frac{2}{n-1} \frac{\partial}{\partial w_{ij}} (u_{ij}^t L^+ L^+ u_{ij}) \\
&= \frac{2}{n-1} (u_{ij}^t \frac{\partial L^+}{\partial w_{ij}} L^+ u_{ij} + u_{ij}^t L^+ \frac{\partial L^+}{\partial w_{ij}} u_{ij}) \\
&= -\frac{2}{n-1} \times (-2) \tau_{ij} L^+ u_{ij} u_{ij}^t L^+ \\
\frac{\partial^2 \bar{\tau}}{\partial {w_{ij}}^2} &= -2 \tau_{ij} \frac{\partial \bar{\tau}}{\partial w_{ij}} \qquad (38)
\end{aligned}
$$

Equation (38) clearly shows that second derivative of $\bar{\tau}$ is non-negative since its first derivative is always negative (according to lemma 4.1) and τ_{ij} is by definition a non-negative function of weights. This completes the proof of lemma 4.1.

References

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

[2] R. Zhang-Shen and N. McKeown. Designing a Predictable Internet Backbone with Valiant Load-Balancing. In *Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005.

[3] L. Valiant and G. Brebner. Universal Schemes for Parallel Communication. In *13th Annual Symposium on Theory of Computing*, May 1981.

[4] C. S. Chang, D. S. Lee, and Y. S. Jou. Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering. In *HPSR '01*, Dallas, May 2001. IEEE.

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

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

[7] P. Van Mieghem and F. A. Kuipers. Concepts of Exact QoS Routing Algorithms. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 12(5):851–864, October 2004.

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

[9] C. Godsil and G. Royle. *Algebraic Graph Theory*. Springer-Verlag, 2001.

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

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

[12] M. Grant and S. Boyd. CVX: Matlab Software for Disciplined Convex Programming (Web Page and Software). $http : //stanford.edu/\ boyd/cvx$. September 2008.

[13] M. Grant and S. Boyd. Graph Implementations for Nonsmooth Convex Programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, $http : //stanford.edu/\ boyd/graph_dcp.html.$ *Lecture Notes in Control and Information Sciences, Springer*, 2008.

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

[15] 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.

[16] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. John Weily and Sons Inc., 1971.