Optimal Design of Signaling Networks for Internet Telephony

Aravind Srinivasan K. G. Ramakrishnan Krishnan Kumaran Murali Aravamudan
Lucent Technologies, Bell Labs Innovations
700 Mountain Avenue, Murray Hill, NJ 07974
{kgr,srin,kumaran,mur}@lucent.com

Abstract—**We present an approach for efficient design of a signaling network for a network of Lucent SoftSwitches supporting Internet Telephony. While one may take an Integer Programming approach to solve this problem, it quickly becomes intractable even for modest-sized networks. Instead, our topology design uses random graphs that we show to be nearly optimal in cost, highly connected, and computationally efficient even for large networks. (Prior work [4] has addressed topology design using random graph techniques. We identified some gaps in this work, for which we provide resolutions.) We then formulate a** *Quadratic Assignment Problem* **(QAP) to map the abstract topology into the physical network to achieve optimal load balancing for given demand forecasts, which we solve using randomized heuristics. Numerical results on several example networks illustrate the performance and computational efficiency of our method. A graphical design tool has been developed based on our algorithms.**

I. Introduction

Efficient design and management of the increasingly complex modern telecommunication networks demands a variety of sophisticated mathematical techniques and algorithms. One such valuable class of techniques, particularly for networks of large size, is based on randomized algorithms and the theory of random graphs [2], [1], [13]. Real-life network design problems often involve hundreds of nodes and hundreds of thousands of demand specifications. An exact integer programming formulation of the problem would involve millions of integer variables and thus be intractable. On the other hand, randomized algorithms possess a rich set of desirable properties, and when applicable, typically require much lower computational complexity compared to deterministic methods, such as those based on optimization. In this study, we describe a network design problem on which randomized algorithms achieve near-optimal solutions. The composite method of using randomized techniques to beat the "curse of dimensionality" in network design problems of realistic size, followed by more specific optimization approaches, thus offers great promise by allowing us to combine the substantial benefits of both techniques.

In this work, we apply these concepts in designing a signaling network for carrying Voice over IP (VOIP). The design is carried out for the deployment of a network of the latest Lucent Technologies Software Switches (abbreviated SoftSwitch, internal project name SARAS). The signaling network for this SoftSwitch network, schematically depicted in Fig. 1, supplants the traditional SS7 signaling network in circuit-switched telephony. The figure shows two switch sites for illustration purposes, but typically will involve about 200 switches in a world-wide network. (The architecture has broader applications, but we stick to Internet telephony terminology here.)

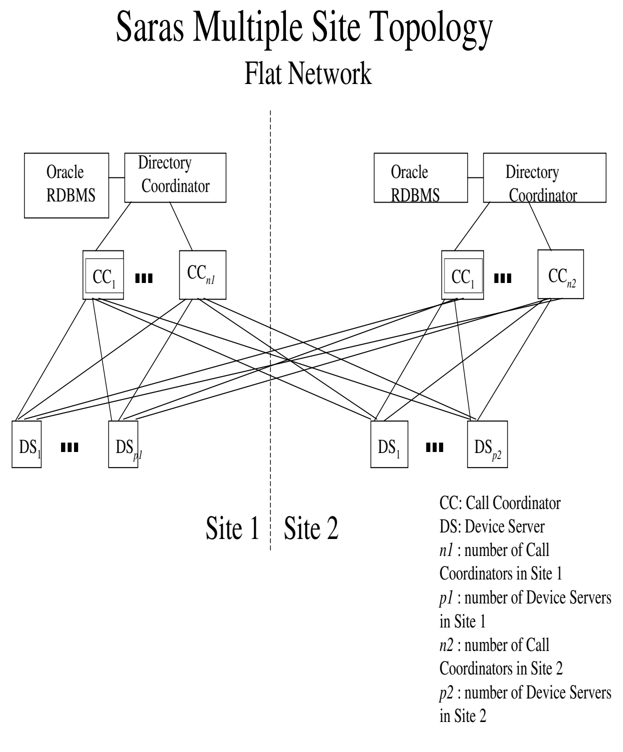

Fig. 1. Lucent SoftSwitch Multiple Site Architecture.

The main components of a SoftSwitch are the device servers (DSs) that process the call setup strings, and the call-coordinators (CCs) that set up the call connections. The DSs talk to the CCs using a proprietary protocol. On a multi-hop call setup the CC in one switch needs to communicate and pass the call setup strings to the DS in the other switch. This communication takes place using sockets and

TCP/IP. Observe that the call setup itself uses the bearer internet. This implies some potential contention between call setup packets and bearer packets, and some mechanism, such as DiffServ or MPLS is needed to assure QoS for the call setup messages. The bearer internet then provides a maximum delay guarantee of about 100ms between any pair of call-coordinators in the network that are directly connected by sockets. Based on this delay guarantee, the signaling network must provide a sufficiently high connectivity and small path-length between call-coordinators, to guarantee acceptable delays in call setup. In this case, it turns out that no call setup signal can traverse more than three call-coordinators, i.e. the origin, an intermediate and the terminating ones, while meeting the tight delay requirements for voice calls.

To minimize the cost of the signaling network, we obtain the minimum number of sockets that achieve the required connectivity. This is our main objective in designing the network topology. The primary constraint is the socket address space constraint, which imposes an upper limit on the number of sockets that each call-coordinator can set up. The problem can be cast as an abstract problem in graph theory, several variants of which have been studied in the literature. Similar problems have been stated and studied in [4]. However, some gaps were identified in the results stated therein, which we remedy here by an alternative approach still based on random graphs.

Once the topology has been designed, one is left with the problem of mapping the nodes onto the graph with the actual switches. Our main consideration here is to do this in a manner so as to load balance the network, i.e., in a manner sensitive to the demands. While we take the demands into account in a limited way in generating the topology, we also formulate an explicit optimization problem to improve the load balancing. The optimization problem takes the form of a Quadratic Assignment Problem (QAP), which is in general NP-hard, and we use randomized heuristics to find good solutions.

The rest of this paper is organized as follows. We first state a general version of the problem and discuss some relevant random graph models. This is followed by a review of the closely related work of [4], in which we point out and resolve certain gaps. We then present our random graph generation technique along with its connectivity properties. This is followed by an augmentation of our method to take the inter-switch demands into account in the random graph generation, after which we present the QAP method to load-balance the network. Following a brief description of a topology design tool that we have developed based on our algorithms, we present numerical results on several example networks, including a real-life network with 65 switches. We end with concluding remarks and the Appendix which proves the paper's theorems.

II. The basic problem

Recent interesting work of [4] has considered related problems of topology design for virtual path (VP) networks, in ATM networking. Given a graph G, recall that the *degree* of a vertex in it is the number of links incident on it; G is called *regular* if all its vertices have the same degree, and *d-regular* if this common degree is d. Also, a *complete graph* has edges between all pairs of distinct vertices.

The basic problem considered in [4] is as follows. One may represent a logical VP network topology by a graph G on n vertices, where each vertex is associated with a network node. An edge in G between vertices u and v, indicates that there is a VP between the corresponding nodes in the network. We are given the number n of vertices, and an upper bound Δ on the maximum degree of any vertex of G. (In our IP telephony context, these links represent sockets, and the degree bound Δ is a socket address-space constraint.) Given n and Δ, the problem is to design G with some desirable characteristics such as:

- **(P1)** G is *k-vertex-connected* for as large a k as possible (i.e., we should delete at least k vertices in order to disconnect G), and
- **(P2)** G has a small *diameter* D (i.e., any two vertices can reach each other within D hops).

Unless G is a complete graph, (P1) is equivalent to the condition that for every pair of distinct vertices u and v, there are at least k pairwise vertex-disjoint paths connecting them. (In practice, we also prefer these paths to be of few hops.)

Remark 1: A graph can be disconnected by removing all neighbors of a vertex. Thus, in (P1), k can be at most as large as the minimum degree of the vertices in G. So, graphs with k equal (or close) to the minimum degree, are good in the sense of (P1).

As discussed in [4], random graphs can be quite fruitful for such network design problems. Let us recall two of the standard models of random n-vertex graphs [2]:

Model $\mathcal{G}_{n,d}$, where we choose a graph uniformly at random from the space of all d-regular, n-vertex graphs; and

Model $G(n,p)$, where we start with n vertices and for each pair of distinct vertices, we put an edge between them with probability p, independently for all vertex-pairs.

Informally, these two models can be considered roughly similar if $d \sim (n-1)p$, since the expected degree of any given vertex in model $G(n,p)$, is $(n-1)p$. But the correspondence is not exact: if n is large enough, then a random graph from $\mathcal{G}_{n,d}$ is connected with high probability if $d \geq 3$, and is disconnected with high probability if $p = p(n)$ has a smaller order of magnitude than the function $(\log n)/n$. The reader is referred to [16] for a recent comprehensive survey of random regular graphs.

The work of [4] advocates $\mathcal{G}_{n,\Delta}$ as an appropriate model for network design problems requiring properties such as (P1) and (P2). Generating a random graph efficiently according to $\mathcal{G}_{n,\Delta}$ can be done in $O(n\Delta^3)$ time, as long as $\Delta \leq O(n^{1/3})$ [11]: no polynomial-time algorithms are known for larger values of Δ. A new algorithm to generate random graphs from a distribution $PM(n, \Delta)$ (introduced in Section II-A) that very closely approximates $\mathcal{G}_{n,\Delta}$, is presented in [4]. In Section II-A, we point out some gaps in this algorithm. In Section II-B, we present some basic difficulties with models $\mathcal{G}_{n,\Delta}$ and $PM(n, \Delta)$ in our network design setting; in Section III, we demonstrate another probabilistic approach that alleviates these difficulties.

A. A gap

Let H be a graph with h vertices. Recall that a *perfect matching* (henceforth "p.m.") in H is a subset of its edges, in which each vertex of H has *precisely one* edge incident on it. (Clearly, a p.m. exists in H only if h is even; if h is even, then a p.m.–if any–has $h/2$ edges.) Let RANDM(H) denote the process of generating a p.m. of H uniformly at random; if H has no p.m., then the process returns nothing. The approach of [4] to construct graphs that possess properties such as (P1) and (P2) with high probability, is as follows. Let H_0 denote the complete graph on n vertices. Run RANDM(H_0), remove the p.m. that is output, and let H_1 denote the remaining graph in which each vertex has $n-1$ links incident. Next run RANDM(H_1), remove the p.m. that is output, and let H_2 denote the remaining graph; this basic step is repeated Δ times. (In other words, H_i is obtained from H_{i-1} by running RANDM(H_{i-1}), and removing the corresponding p.m. from H_{i-1}.) Finally, we output the union of the Δ perfect matchings found. Let $PM(n, \Delta)$ denote this random graph model; clearly, any graph generated from this model is Δ-regular.

The following randomized procedure is suggested in [4] for RANDM(H), for graphs H that have at least one p.m. Generate a random permutation of the edges of H, and assign a weight of 2^i for the ith edge in this permutation. Now compute a p.m. of maximum total weight (this can be accomplished in $O(h^3)$ time [6]). It is claimed in [4] that this is a correct implementation of RANDM(H); i.e., that every p.m. is equally likely to occur. We show that this claim is unfortunately false, through graphs H_k of the form shown in Example 1 of Figure 3, where k is any even integer. Note that H_k has $2k+4$ vertices: u, v, w, x, and two complete graphs on k vertices, A and B. (That is, each of A and B has all possible edges among its k elements.) Each of u and w is connected to all vertices of A; each of v and x is connected to all vertices of B. Further, we have edges $e_1 = \{u, v\}$ and $e_2 = \{w, x\}$. H_k is $(k+1)$-regular; the total number of edges is $m = (k+1)(k+2)$. Define a p.m. in H_k to be of *Type A* iff it contains both of the edges e_1 and e_2; define a p.m. in H_k to be of *Type B* iff it contains neither of the edges e_1 and e_2. Since k is even, a simple argument shows that any p.m. in H_k is either of Type A or of Type B. Now, the number of p.m.s in a complete graph with an even number ℓ of vertices is $f(\ell) \doteq (\ell-1)(\ell-3)\cdots 1$. (Define $f(0) = 1$ for convenience.) Thus, we can check that H_k has $(f(k))^2$ p.m.s of Type A, and $(k(k-1)f(k-2))^2$ p.m.s of Type B.

Recall that $m = (k+1)(k+2)$ is the number of edges in H_k. Let $\mathcal{E}$ be the event that either e_1 or e_2 is assigned weight 2^m; clearly, $\Pr[\mathcal{E}] = 2/m = 2/((k+1)(k+2))$. Now, as pointed out in [4], since $\sum_{i=1}^{m-1} 2^i < 2^m$, the event $\mathcal{E}$ will force a p.m. of Type A to be chosen. Thus, since there are $(f(k))^2$ p.m.s of Type A, there exists a p.m. M_0 of Type A, such that the probability of M_0 being output is *at least* $p_0 = \frac{2}{(k+1)(k+2)(f(k))^2}$. However, since there are $(f(k))^2 + (k(k-1)f(k-2))^2 = (1+k^2)(f(k))^2$ p.m.s in total, RANDM(H_k) requires each p.m. to be output with probability $p = [(1+k^2)(f(k))^2]^{-1}$. Noting that $p_0 \neq p$ (and in fact that for large k, p_0 is almost twice as large as p), we see that the above randomized algorithm does not correctly implement RANDM(H). In fact, there is a Type A matching that is output with probability roughly $k/2$ times the required value of p:

Theorem 1: There is a Type A matching that is output with probability at least $(1+k^2)p/(2k+1)$ ($\sim (k/2)\cdot p$ for large k).

Theorem 1 is proved in the appendix. Furthermore, there are also *bipartite* graphs such as those of Example 2 in Figure 3, on which the algorithm is not correct. (The error follows from a claim in Lemma 1 of [4]; the reader is referred to the notation and proof approach of Lemma 1 in [4]. Towards the end of the proof, it is claimed that the weight "2^j" is equally likely to fall in $M_1 - M_2$ or in $M_2 - M_1$. While this is true if we do not condition on anything, what the proof really needs is this, *conditional on* the event "$A+B$". Unfortunately this conditional statement is not true.)

B. Some further difficulties

Even if an efficient implementation of RANDM(H) is discovered, the proof that $PM(n, \Delta)$ and $\mathcal{G}_{n,\Delta}$ closely correspond to each other [12], seems to be for *constant* values of Δ. Thus, it is unclear if this general approach works for arbitrary $\Delta = \Delta(n)$. One way to get around these is to return to the $\mathcal{G}_{n,\Delta}$ model, and employ the following beautiful algorithm of [8]. Let $N(n,\Delta)$ denote the set of Δ-regular n-vertex graphs, with $|N(n,\Delta)|$ denoting its cardinality. Given any parameter $\epsilon > 0$, the algorithm of [8] generates an element of $N(n,\Delta)$; for each given $H \in N(n,\Delta)$, its probability of generation lies between $|N(n,\Delta)|^{-1}/(1+\epsilon)$ and $|N(n,\Delta)|^{-1}(1+\epsilon)$. Thus, for *any* event $\mathcal{A}$, if q denotes its probability under the $\mathcal{G}_{n,\Delta}$ model, the probability of $\mathcal{A}$ when using this algorithm lies in the interval $[q/(1+\epsilon), q(1+\epsilon)]$. In particular, high-probability events (such as properties (P1) and (P2)) under $\mathcal{G}_{n,\Delta}$, remain so if we employ this algorithm of [8]. However, though this algorithm runs in time polynomial n and $1/\epsilon$, the time is somewhat high: at least $\Omega(n^5)$. Moreover, it is mentioned in [4] that a random graph chosen from $\mathcal{G}_{n,\Delta}$ is almost surely Δ-vertex-connected. (Here and henceforth, "almost surely" means with probability tending to 1 as n increases.) While this is true for all constant Δ, this is currently only proven for $3 \leq \Delta(n) = O(n^{0.02})$ [10]: see the discussion in pages 12 and 13 of [16].

Thus, there are a few bottlenecks with direct use of models such as $\mathcal{G}_{n,\Delta}$ and $PM(n,\Delta)$ in our network design setting; Section III presents an approach that gets over these.

III. Using $G(n,p)$ for topology design

The case where, say, $\Delta \leq n^{0.02}$ can be handled via the $O(n\Delta^3)$ time algorithm of [11] to generate a random graph according to $\mathcal{G}_{n,\Delta}$. As mentioned above, work of [10] shows that in this case, the generated graph is almost surely Δ-connected. Also, graphs drawn from $\mathcal{G}_{n,\Delta}$ almost surely have nearly best-possible diameter D for graphs of maximum degree Δ [4], [16]: for any fixed $\rho > 0$, we almost surely have $1 + \lfloor \log_{\Delta-1} n \rfloor + \lfloor \log_{\Delta-1}(\frac{\Delta-2}{6\Delta} \log n) \rfloor \leq D$, and

$$D \leq 1 + \lfloor \log_{\Delta-1}((2+\rho)\Delta n \log n) \rfloor. \quad (1)$$

As mentioned in [4], D cannot be much smaller than the lower-bound on D given implicitly by (1); the *Moore bound* shows that

$$n \leq 1 + \Delta((\Delta-1)^D - 1)/(\Delta-2) \leq 1 + \Delta^D. \quad (2)$$

For instance, it is shown in [4] that if Δ grows with n, then for n large enough, the upper bound (1) is to within $1+\epsilon$ of the lower bound on D given by (2), for any fixed $\epsilon > 0$. Thus, the case where $\Delta \leq n^{0.02}$ can be handled using these known methods; the running time for the topology generation is $O(n\Delta^3) \leq O(n^{1.06})$. So, we shall focus on the case $\Delta > n^{0.02}$. Also, as mentioned in [4], these network design problems become more challenging as the number n of nodes increases (the "curse of dimensionality"). Thus, we shall focus on the case where n is large. For "small" values of n, Section IV presents certain heuristics that work well in conjunction with random graphs.

For $\Delta > n^{0.02}$, we propose generating a random graph according to $G(n,p')$, where $p' = \Delta(1-\epsilon)/(n-1)$ with $\epsilon = \sqrt{3(\ln n)/\Delta}$; note that ϵ is vanishingly small as n increases, since $\Delta > n^{0.02}$. Thus, the random graph generation here takes only $O(n^2)$ time. We will show that this approach also brings us desired properties such as degrees bounded by Δ, (P1), and (P2). For the sake of generality, we prove such a result for any $p = p(n) \geq (\ln n)^6/n$ in Theorem 2; our desired results will be direct corollaries. The proof of Theorem 2 is shown in the Appendix.

We define $(m)_r = m(m-1)(m-2)\cdots(m-r+1)$. As usual, let e denote the base of the natural logarithm; $\exp(x)$ denotes e^x. We first present a useful fact:

Fact 1: For all large enough n and any integer $L' \leq \lceil (\ln n)/\ln\ln n \rceil$, $(n-2)_{L'-1} \geq n^{L'-1}\exp(-2(L')^2/n)$.

Proof: By calculus, $1 - x \geq \exp(-2x)$ holds for $0 \leq x \leq 1/2$. Also, if n is large enough, $i/n \leq 1/2$ for all $i \leq L'$. Thus, $(n-2)_{L'-1} \geq n^{L'-1}\exp(-2\sum_{i=2}^{L'} i/n)$, which is at least $n^{L'-1}\exp(-2(L')^2/n)$. ■

Theorem 2: Let $\rho \in (0,1]$ be any constant, $p = p(n) \geq (\ln n)^6/n$, and n be sufficiently large. Then, with probability tending to 1 as n increases, a random graph H generated from $G(n,p)$ satisfies all of the following properties.

(a) The degrees of all vertices lie in the range $[(n-1)p(1-\delta), (n-1)p(1+\delta)]$, where $\delta = \sqrt{\frac{(2+\rho)\ln n}{(n-1)p}}$.

(b) Let L be the smallest integer such that $(n-2)_{L-1}p^L \geq (2+\rho)\ln n$. Then, H has diameter at most L.

(c) H is $(n-1)p(1-2\delta)$-vertex-connected.

Since δ is vanishingly small, part (a) basically shows that all vertex degrees are close to their expected value of $(n-1)p$. As δ is tiny, Remark 1 shows that (c) is essentially optimal.

As for (b), a more general result is proved in [2]; e.g., for values such as L, the probability that the diameter *equals* L, is determined asymptotically (as $n \to \infty$) there. We present a shorter proof to derive our result.

We simplify (b) by showing that $L = \lceil (\ln n)/\ln\ln n \rceil$, say, satisfies $(n-2)_{L-1}p^L \geq (2+\rho)\ln n$. By Fact 1, it suffices to show that

$$(np)^L \exp(-2L^2/n) \geq (2+\rho)n\ln n. \quad (3)$$

Since $np \geq (\ln n)^6$, it is easy to check that (3) holds for our value of L, for all large n.

Often, a network design mandates a diameter bound: i.e., the diameter is required to be at most some given L. (For instance, a condition very similar to "$L = 2$" is required in our Internet telephony design discussed in Section IV. Also, no value p satisfies part (b) of Theorem 2, if the given diameter bound L is 1. But if $L = 1$, we know what the topology should be: the complete graph on n vertices.) From above, our range of p can be applied directly if $L > (\ln n)/\ln\ln n$; so suppose $L \leq O((\ln n)/\ln\ln n)$. Bound (3) shows that as n increases, $p \geq 1.8(n\ln n)^{1/L}/n$, say, suffices; so, by (a), the maximum degree Δ is at most, say, $2(n\ln n)^{1/L}$. Rearranging, we see that

$$L \leq \frac{\ln n + \ln\ln n}{\ln\Delta - \ln 2}. \tag{4}$$

Now, the Moore bound (2) shows that $L \geq \ln(n-1)/\ln\Delta$. As argued in [4], it is easy to check that our upper bound on L from (4), approaches this lower bound on L as $n \to \infty$, since $\Delta \to \infty$ as $n \to \infty$. Thus, (b) is nearly best-possible.

Recall that we use $G(n,p)$ with $p = p' = \Delta(1-\epsilon)/(n-1)$, where $\Delta > n^{0.02}$ and n is large. So, the above discussion shows that the resulting network is almost surely near-optimal in terms of the relevant parameters of maximum degree, diameter, and vertex-connectivity. Finally, the condition "$p(n) \geq (\ln n)^6/n$" of Theorem 2 is sufficient for our purposes, since we only require $p \geq n^{0.02}/n$. This lower bound of $(\ln n)^6/n$ on $p(n)$ can be decreased to about $\Omega((\ln n)/n)$, via a more detailed analysis; this is omitted here for lack of space.

IV. SIGNALING NETWORK DESIGN

We now describe key issues in the design of the signaling network for Lucent's SoftSwitch. Abstracting away those issues that are not germane to this discussion, the network design problem is as follows. We have s software switches, each having some number cc of *call-coordinators*. These call-coordinators considerably help in parallel processing, among other benefits; given a call to be setup from one switch to another, any pair of call-coordinators in these respective switches can take up the task. The parameter cc in our practical setting is of the order of a few tens, and is at most 50. The parameter s can vary up to about a few hundreds: under traffic conditions predicted for the next several years, a design with $s = 200$ and $cc = 50$, say, will suffice to cover Internet telephony worldwide.

The goal here is optimal design of a signaling network G interconnecting the call-coordinators (and hence the switches). Note that the "links" in this network are TCP/IP sockets. For convenience, we will often refer to the call-coordinators as nodes or vertices. Note that G has $n = s \cdot cc$ vertices. A first constraint $\mathcal{C}_1$ is that there be no links between call-coordinators in the same switch. For reasons of fault-tolerance, the main constraint $\mathcal{C}_2$ is that any call-coordinator v_1 from any switch sw_1 should be reachable from any call-coordinator v_2 of every *other* switch sw_2, within two hops. Thus, recalling $\mathcal{C}_1$, if there is no direct link (socket) between v_1 and v_2, there should be a path of length two between them, whose intermediate vertex (call-coordinator) lies neither in sw_1 nor in sw_2. Hence we see that for each pair of distinct switches sw_1 and sw_2, a call can be setup between these within two hops, as long as *any one* call-coordinator from both of these switches is functioning. Note that this condition is closely related to G having diameter 2, but is not exactly the same. Due to socket address-space constraints, the next main constraint is that the maximum degree of any vertex (call-coordinator) in G be as small as possible; if there are many such low-degree designs, we aim for one that has the least number of total links. A next-level goal is to "load-balance" the network: loads should be relatively evenly carried, and high-demand pairs of switches should be connected by as many direct links as possible. In general, there should be several short switch-disjoint paths between switch-pairs. We aim to provide methods of low computational complexity that provide good designs while incorporating demands between node pairs in a natural manner.

As mentioned above, $\mathcal{C}_2$ is not exactly that G have diameter 2 with small maximum degree Δ, but is close enough that we can apply the Moore bound argument to lower-bound Δ. Consider any call-coordinator v_1 in any switch sw_1. Its number of neighbors is at most Δ, and number of vertices at distance two from it is at most $\Delta(\Delta-1)$. These vertices should cover all call-coordinators lying outside sw_1; i.e., $\Delta + \Delta(\Delta-1) \geq (s-1)cc$. Thus, $\Delta \geq \sqrt{n - cc}$; so, Δ is lower-bounded by roughly $\sqrt{n}$.

To enforce $\mathcal{C}_2$ with Δ not being large, a randomized algorithm very similar to $G(n,p)$ can be shown to work well: independently for every pair of call-coordinators in *different* switches, put a link between them with probability $p \sim \sqrt{2(\ln n)/n}$. This will also ensure with high probability that $\Delta \sim d \doteq \sqrt{2n\ln n}$. In practice, for moderate values of n, we found the following alternative to yield slightly better values for Δ. Take d' slightly smaller than $d/2$, such as $\lfloor 0.45d \rfloor$. First generate d' random permutations $\{\pi^{(1)}, \pi^{(2)}, ..., \pi^{(d')}\}$ of the nodes. Next, connect node i to nodes $\pi_i^{(\ell)}$ for $\ell \in \{1, 2, .., d'\}$. (If $i = \pi_i^{(\ell)}$, the link $\{i, \pi^{(\ell)}\}$ is not added.) One further advantage is that the maximum degree here is at most $2d'$, *with probability one*. Since $2d' < d$, this would lead to a smaller value for the maximum degree, but will likely violate $\mathcal{C}_2$. We then process the pairs of call-coordinators that are at a distance more than two, in *random order*: when it is the turn of such a pair, we add a link between them if necessary. The benefit is that for moderate values of n, the final maximum degree is typically smaller than d. Our algorithm **UNIF**, augmented by such heuristics, leads to improved values for the maximum degree and total number of sockets for the design problems that arise in practice.

A. Incorporating Demands

The above algorithm **UNIF** assumes homogeneous demands between switch-pairs. We meld it with a demand-sensitive approach as follows. Suppose we are given a symmetric demand matrix d_{ij}, representing the aggregate demand for call set-ups between call-coordinators i and j for both directions. This demand is obtained by dividing the inter-switch demands equally among the call-coordinator pairs. In this context, we have assumed the sockets to be bi-directional, i.e., fully duplex, although our methods can be extended to the asymmetric case with additional effort using directed graphs. Returning to the symmetric case, the following algorithm can be shown to provide a graph with average degree Δ with a demand-sensitive degree variation among the nodes.

Algorithm DEM:

1. For each ordered pair of nodes (i, j) from different switches, independently place a directed link from i to j with probability $p_{ij} = \frac{\Delta}{2(s-1)cc}\frac{d_{ij}}{\sum_k d_{ik}}$.

2. Replace all directed links by undirected (bidirectional) links. (Also, if directed links (i, j) and (j, i) are both present, they are replaced by the single undirected link $\{i, j\}$.)

It is easy to see that the expected out-degree of each node after step 1 is $\Delta/2$. By the proof approach for part (a) of Theorem 2, it can also be shown that all these out-degrees are close to $\Delta/2$ with high probability; in particular, after step 2, the average degree of the vertices will be close to Δ. But some indegrees may become large after step 1 (e.g., if vertex i is such that for all j and k different from i, $d_{ji} \gg d_{jk}$). Moreover, the demands may be such that with good probability, $\mathcal{C}_2$ is *not* satisfied: consider, e.g., an extreme case where there are two disjoint sets of switches S_1 and S_2 where intra-switch demand within each of S_1 and S_2 is high, while inter-switch demand between S_1 and S_2 is low. In such a case, a purely demand-based approach may even disconnect S_1 from S_2. However, the demand-sensitive feature of **DEM** is attractive. Thus, we developed the following combination of algorithms **UNIF** and **DEM**.

Algorithm COMBO

Suppose our target degree is approximately Δ. For each pair (Δ_1, Δ_2) of non-negative integers such that $\Delta = \Delta_1 + \Delta_2$, perform the following steps.

- Run algorithm **UNIF** to generate a graph with maximum degree roughly Δ_1.
- Run algorithm **DEM** to generate a graph with maximum degree approximately Δ_2.
- Superimpose the above two graphs.

Finally, we pick the pair (Δ_1, Δ_2) that leads to the "best" superimposed graph (e.g., having a small maximum degree and a small total number of sockets; among the possibly many equally good choices here, choose the one that is most demand-sensitive).

B. Load Balancing the Network

We now discuss a measure to quantify the extent of load balancing in the network once the topology has been determined using the above described methods. This measure may then be used as an additional criterion to choose a topology from among multiple feasible choices. In the absence of limitations on link capacities, as is our case, we aim mainly to load balance among the various paths in the network, keeping the traffic per path as low as possible. To this end, we define the following connectivity measure between a given pair of switches (i, j) : $c_{ij} = \alpha\mathbf{1}_{ij} + \mathbf{2}_{ij}$ where $\mathbf{1}_{ij}$ is the number of direct edges and $\mathbf{2}_{ij}$ is the number of 2-hop paths between call-coordinators in switches i and j respectively. α is a weighting factor indicating the relative importance of direct edges over 2-hop paths, included in anticipation of the fact that direct edges can afford to accommodate greater traffic on account of higher free capacities and smaller delays. Our load balancing measure is then defined as

$$\mathcal{L} = \sum_{i,j\neq i} c_{ij}/d_{ij}. \tag{5}$$

$\mathcal{L}$ is a measure of the average number of paths per unit of demand, and hence picking choices with larger values of this measure ensures that switch pairs with high demands are provided greater path diversity. This also provides a certain degree of reliability against failures. We hence use $\mathcal{L}$ to decide among feasible design choices with equal or comparable costs in terms of total number of sockets.

In addition to the above, the load balancing measure may be used to map the abstract nodes of the graph from algorithm **UNIF** into the physical switches. This is motivated by the fact that we have observed appreciable variation in c_{ij} in the graphs produced by **UNIF**. Also in algorithm **COMBO**, while the demand based random graph generation implicitly starts with such a mapping, it is often the case that the final solution is heavily dominated by the Δ_1 component owing to the tight degree constraints. To further explain this process, it is necessary to make the distinction between the abstract topology, namely the output graph from the previously described algorithms, and the physical switches. Since we are now dealing at the switch level, our subsequent use of the words "node/vertex" really refer to sets of nodes/vertices corresponding to call coordinators belonging to a single switch. If we number these nodes on the graph as $\{1, 2, ..., s\}$, where s is the number of switches, a mapping is then a permutation π of this set. It is then clear that the load measure (5) takes the form

$$\mathcal{L}(\pi) = \sum_{i,j \neq i} \frac{c_{ij}}{d_{\pi_i \pi_j}}. \tag{6}$$

In terms of the binary indicator variables x_{ip}, defined to be 1 if node i is mapped to switch p and 0 otherwise, the above equation can be rewritten as

$$\mathcal{L}(\mathbf{x}) = \sum_{i,p,j \neq i, q \neq p} \frac{c_{ij}}{d_{pq}} x_{ip} x_{jq}. \tag{7}$$

The "optimal" mapping, from the load-balancing point of view, is then defined to be the choice $\mathbf{x}$ that maximizes (7) subject to the standard assignment constraints $\sum_i x_{ip} = 1 \; \forall p$, and $\sum_p x_{ip} = 1 \; \forall i$. This formulation corresponds to the well-known NP-hard combinatorial problem known as the *Quadratic Assignment Problem* (QAP) (see [15]), for which efficient heuristics are nevertheless available. In particular, we use a technique known as as GRASP [5] (acronym for Greedy Randomized Adaptive Search Procedure) which has been successfully applied to other similar optimization problems.

To obtain an upper bound on the best possible solution to this problem, we invoke a Linear Programming (LP) based technique that we hereafter term as the "LP relaxation". The above quadratic program can be converted to a linear program using the larger set of variables $y_{ipjq} = x_{ip} x_{jq}$ in equation (7), along with the additional constraints,

$$\sum_{j \neq i} y_{ipjq} = x_{ip} \; \forall q \neq p; \; \sum_{q \neq p} y_{ipjq} = x_{ip} \; \forall j \neq i. \tag{8}$$

for all i, p. The word "relaxation" refers to relaxing the integrality constraints on the variables and allowing $0 \leq y_{ipjq} \leq 1$. When the number of switches is small (less than 50), the linear program can be solved using Interior Point techniques [9], and necessarily produces an upper bound on the best possible QAP solution. We report on the results of these calculations on some realistic examples subsequently.

V. SARASWATI: A Topology Design Tool

We have incorporated Algorithm COMBO into a graphical design tool, SARASWATI. SARASWATI is an acronym for SARAS Wide Area Topology optImizer. The software tool runs on WINDOWS 95/98/NT as well as UNIX platforms. The tool has two major software modules: a graphical user interface and a backend optimizer implementing COMBO. The GUI, written in TCL/Tk language [14] lets

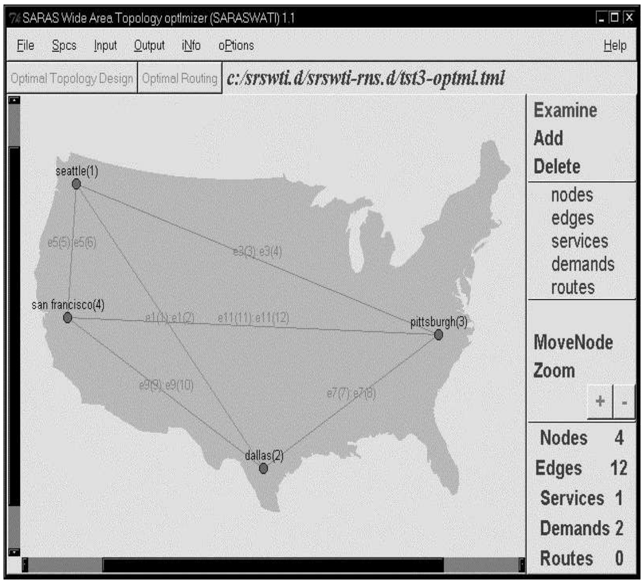

Fig. 2. Snapshot of the Initial canvas, with switch sites drawn

the user graphically place the location of the switches on a background US map, using the widgets on the right of the canvas. A snapshot of the canvas is shown in Fig. 2. The other main component of the input specification, the demands between the switches, can also be accomplished by the widgets on the right. However, if the network to be optimized has hundreds of switches and tens of thousands of demands, it is more convenient to describe the demands in a textual form. We have designed a simple non-procedural language to describe the demands. The user can save the current network drawn as an ascii file which encodes the problem in this language; the user can also load existing problem files described in this language. After the problem is completely described, the user loads a specification file, stating various additional parameters, such as the number of call coordinators per switch, the socket address space constraints, etc. The user then clicks on the menu "optimal topology design" to invoke the optimizer. After solving the topology design, the user is given the option to view the solution. The user can also view the summary statistics of the topology by using the *output* menu. We are currently designing a client-server version of the tool which can then be used in a Web environment.

VI. Numerical Results

We now illustrate our design methods on several example networks of various sizes, including some realistic ones. A key point is that in all of our runs (including the 12 examples shown below), the number of switch-disjoint paths of length at most two between *any* pair of switches, is the maximum possible value of $s - 1$; thus, we achieve maximum vertex-connectivity (in fact by considering only paths of length at most two). The results for the random topology generation using algorithm **COMBO** are summarized in table VI, using the following notation:

- # : Experiment number used for future reference.
- Input data:
  - s = Number of switches.
  - $n = s \times cc$ = Total number of nodes in graph.
- Algorithm Outputs:
  - Δ^{max} = Maximum degree of a call coordinator.
  - Δ^{ave} = Average degree; this is our objective value, since the total number of sockets is given by $n * \Delta^{ave}/2$.
  - Δ^{min} = Minimum degree.

#	s	n	Δ^{max}	Δ^{ave}	Δ^{min}
1	10	500	52	46.51	40
2	20	1000	75	68.90	60
3	30	1500	93	86.93	76
4	40	2000	109	101.86	89
5	50	2500	124	115.86	101
6	100	5000	174	168.54	157
7	200	10000	257	247.77	241

TABLE I

RESULTS OF RANDOM TOPOLOGY GENERATION.

For these experiments, the number of call coordinators per switch (cc) was set at 50, a choice based on an existing design, and the pair-wise demands d_{ij} were generated uniformly at random in the range $[1, 100]$. Observe that Δ^{ave} is at most $2.5\sqrt{n}$, with degree variance of a few %. Some of this variation is on account of sensitivity to the demands, while some is due to random fluctuations. As mentioned before, the random graph procedure is of low complexity and generates results for the $n = 10000$ case within an hour or so on a reasonably fast workstation. An integer programming based formulation of this problem would be unapproachable using existing techniques due to variable explosion and computational complexity. Also, as mentioned above, we always get the best-possible vertex-connectivity of $s - 1$. It is possible to satisfy the design constraint $\mathcal{C}_2$ with networks of maximum degree about $\sqrt{n}$ (which is close to the lower bound of $\sqrt{n - cc}$), but these constructions do not have sufficiently high vertex-connectivity at the switch level.

Following this, we used the QAP-based load balancing method described earlier, and the results are summarized in table VI. The notation is as follows:

- $\mathcal{L}^0$ = Objective value with originally assumed random matching.
- $\mathcal{L}^{GRASP}$ = Objective value obtained from best matching produced by our GRASP heuristic.
- $\mathcal{L}^{LP}$ = Upper bound for best possible objective value based on the linear programming relaxation of quadratic assignment.
- $r = 100\left[\frac{\mathcal{L}^{GRASP}}{\mathcal{L}^0} - 1\right]$ = % Improvement in load balancing from our GRASP heuristic
- $e = 100\left[\frac{\mathcal{L}^{LP}}{\mathcal{L}^{GRASP}} - 1\right]$ = % Difference between GRASP result and best possible solution from LP relaxation.

Expt. No.	r	e
1	2.83	0
2	1.5	0.02
3	1.2	0.15
4	1.4	0
5	1.4	-
6	0.9	-
7	0.75	-

TABLE II

RESULTS OF QAP LOAD BALANCING PROBLEM.

Observe that the QAP solutions are about 0.75-3% better load balanced than the originally assumed mappings, from the values of r. For the larger examples, with 50 switches or more, the LP upper bound is too burdensome to evaluate, but the high efficiency of the GRASP heuristic is evident from the values of e in the smaller cases.

In our final series of experiments, we studied a real example of a network with 65 switches and actual measured demands between switch pairs. Table VI summarizes the quality of the designs with differing numbers of call coordinators per switch. This study helps in designing a least cost signaling network subject to the constraint that the load on each call coordinator is at an acceptable level. The notation is as described before. Note that the improvement in the load balancing is 3-6%, which could result in a similar reduction in the number of call coordinators and sockets, and hence the cost of the network.

#	cc	Δ^{max}	Δ^{ave}	Δ^{min}	r
1	10	56	54.35	52	6.7
2	20	86	80.38	73	4.9
3	30	106	100.41	92	2.7
4	40	126	117.92	104	5.0
5	50	142	133.50	119	3.6

TABLE III

RESULTS OF ALGORITHM **COMBO** AND QAP ON THE 65 SWITCH NETWORK WITH VARYING cc.

VII. Conclusions

We have presented novel techniques based on random graphs to design networks with low degree, high connectivity and good load balancing properties. Our methods, which address the specific purpose of designing Internet Telephony signaling networks, are illustrated on several random examples as well as a real AT&T network example. In a more general sense, we propose that the combination of randomization followed by the use of more specific optimization techniques is an effective one for many large scale network design problems. This work serves to illustrate that fact by first designing the topology at the call-coordinator level, and performing load balancing at the much coarser switch level.

Possible extensions include incorporating various forms of asymmetry into the randomized procedure - such as directed links - and performing network design by accounting for other limitations such as link capacity constraints.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method.* John Wiley & Sons, Inc., New York, 1992.

[2] B. Bollobás. *Random Graphs*. Academic Press, London, 1985.

[3] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[4] A. Faragó, I. Chlamtac, and S. Basagni. Virtual Path Network Topology Optimization Using Random Graphs. In *Proc. of IEEE INFOCOM'99*, pages 491–496, 1999.

[5] T. A. Feo, M. G. C. Resende, and S. H. Smith. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research*, 42(5):860–878, 1994.

[6] H. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. ACM*, 23:221–234, 1976.

[7] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

[8] M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *Theoretical Computer Science*, 73:91–100, 1990.

[9] N. Karmarkar and K. Ramakrishnan. Computational Results of an Interior Point Algorithm for Large Scale Linear Programming. *Mathematical Programming*, 52:555–586, 1991.

[10] T. Łuczak. Sparse random graphs with a given degree sequence. In A. Frieze and T. Łuczak, editor, *Random Graphs, Vol. 2*, pages 165–182. Wiley, New York, 1992.

[11] B. D. McKay and N. C. Wormald. Uniform generation of random regular graphs of moderate degree. *J. Algorithms*, 11:52–67, 1990.

[12] M. S. O. Molloy, H. Robalewska, R. W. Robinson, and N. C. Wormald. 1-factorizations of random regular graphs. *Random Structures & Algorithms*, 10:305–321, 1997.

[13] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[14] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[15] P. Pardalos and H. Wolkowisz (Eds.). *Quadratic Assignment and Related Problems*. American Mathematical Society, New York, 1994.

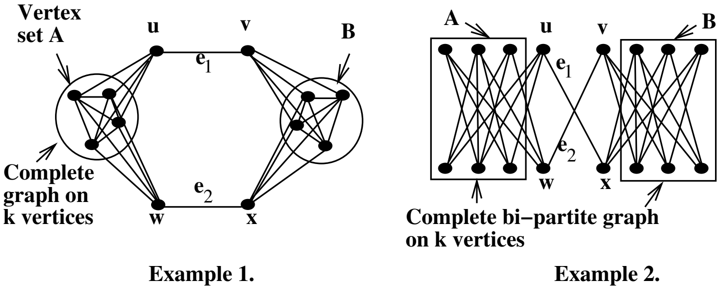

Fig. 3. Examples for theorem 1.

[16] N. C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics*. 1999. In press. Available from *http://www.ms.unimelb.edu.au/∼nick/papers.html.*

Appendix: Proofs of Theorems

Proof of Theorem 1. In H_k, call the $4k + 2$ edges incident on u, v, w, x as *special*; the $4k$ special edges excluding e_1 and e_2, are called *ordinary*. Consider the event $\mathcal{F}$: "either e_1 or e_2 gets maximum weight among the special edges". We claim that if $\mathcal{F}$ happens, then a Type A p.m. will be of maximum weight. Indeed, consider any weight assignment for the edges in which $\mathcal{F}$ holds, and let M^* be any Type B p.m. M^* will have $k - 2$ non-special edges and four ordinary edges, one each incident on u, v, w, x: say, edges $\{u, u'\}$, $\{v, v'\}$, $\{w, w'\}$ and $\{x, x'\}$. We show that M^* cannot be of maximum weight, by constructing a Type A p.m. M' of greater weight. M' has e_1, e_2, the non-special edges of M^*, and the edges $\{u', w'\}$, $\{v', x'\}$. Since $2^i > \sum_{j=1}^{i-1} 2^j$ for any positive integer i and since event $\mathcal{F}$ holds, we can check that M' has more weight than M^*. Thus, a Type A p.m. will be of maximum weight if $\mathcal{F}$ happens. So, since $\Pr[\mathcal{F}] = 2/(4k + 2) = 1/(2k + 1)$ and there are $(f(k))^2$ Type A p.m.s, some p.m. of Type A is chosen with probability at least $[(2k + 1)(f(k))^2]^{-1}$. □

Before proving Theorem 2, we start with some useful tools. Let $[k]$ denote the set $\{1, 2, \ldots, k\}$, and $\overline{A}$ denote the complement of an event A. As usual, the Boolean connectives "and" and "or" are denoted respectively by $\wedge$ and $\vee$. "Random variable" is abbreviated as "r.v.".

Chernoff-Hoeffding bounds [3], [7]. Let $X_1, X_2, \ldots, X_\ell$ be *independent* r.v.s, each taking values in $[0, 1]$, with $R = \sum_{i \in [\ell]} X_i$ and $\mathbf{E}[R] \le \mu$. For any $\beta \ge 0$,

$$\Pr(R \ge \mu(1 + \beta)) \le F_1(\mu, \beta) \doteq (e^\beta/(1 + \beta)^{(1+\beta)})^\mu; \quad (9)$$

if $\mathbf{E}[R] \ge \mu$ and $0 \le \beta < 1$,

$$\Pr(R \le \mu(1 - \beta)) \le F_2(\mu, \beta) \doteq e^{-\mu\beta^2/2}. \quad (10)$$

A useful fact about the function $F_1(\mu, \beta)$:

$$F_1(\mu, \beta) \leq \exp(-(\beta^2/2+\beta^3/6)\mu) \text{ if } \mu \geq 0 \text{ and } \beta \in [0,1]. \tag{11}$$

To prove this, we take logarithms and aim to show that $f(x) = (1+x)\ln(1+x)-x+x^3/6-x^2/2$ is non-negative, for $x \in [0,1]$. To see this, note that the third derivative $f'''(x)$ equals $1 - 1/(1+x)^2$, which is non-negative for $x \in [0,1]$. Since $f''(0) = 0$, this shows that $f''(x) \geq 0$ for $x \in [0,1]$. Since $f'(0) = 0$, this yields $f'(x) \geq 0$ for $x \in [0,1]$. Finally, as $f(0) = 0$, $f(x) \geq 0$ for $x \in [0,1]$.

We next present **Janson's inequality** (Theorem 1.1 in Chapter 8 of [1]). Suppose $Z_1, Z_2, \ldots, Z_\ell \in \{0,1\}$ are *independent* random variables. Let $A_1, A_2, \ldots, A_s$ be some given subsets of $[\ell]$, and B_i be the event "$\forall j \in A_i,\ Z_j = 1$". Janson's inequality gives good upper bounds on $\Pr[\bigwedge_{i\in[\ell]} \overline{B_i}]$, under certain conditions specified now. For $i, j \in [\ell]$, we say $i \sim j$ ("i is related to j") iff $i \neq j$ and $A_i \cap A_j \neq \emptyset$. Define $\Lambda = \sum_{(i,j):i\sim j} \Pr[B_i \wedge B_j]$, the sum being taken over *ordered* pairs (i,j). Suppose $\Pr[B_i] \leq \nu$ for all i, and that $M = \prod_{i\in\ell} \Pr[\overline{B_i}]$. Then, Janson's inequality shows that

$$\Pr[\bigwedge_{i\in[\ell]} \overline{B_i}] \leq M \exp(\Lambda/(2(1-\nu))). \tag{12}$$

Finally, suppose h is a function mapping graphs to {True, False}. Call h increasing if, for graph H such that $h(H)$ is True, $h(H')$ is also True for any graph H' (with the same vertex-set as A) that is obtained by adding zero or more edges to H. For instance, if $h(H)$ is True iff H has diameter at most k (where k is some given constant), then h is clearly increasing. Let $h_{n,p}$ be the probability that for a graph H drawn randomly from $G(n,p)$, $h(H)$ is True. Theorem 1 in Chapter 2 of [2] shows the intuitive fact that for any increasing h,

$$\forall n,\ h_{n,p_1} \leq h_{n,p_2} \text{ if } p_1 \leq p_2. \tag{13}$$

Proof of Theorem 2. We label the vertices by $1, 2, \ldots, n$. For all unordered pairs $\{i,j\}$ of distinct elements of $[n]$, define $Y_{i,j}$ to be 1 if H contains the edge $\{i,j\}$, and let $Y_{i,j} = 0$ otherwise. (The random variables $Y_{i,j}$ and $Y_{j,i}$ are identical.) Note that all the $\binom{n}{2}$ random variables $Y_{i,j}$ are *independent*, and have $\Pr[Y_{i,j} = 1] = p$.

(a) Fix any vertex i. The degree D_i of i in H equals $\sum_{j\neq i} Y_{i,j}$. By the linearity of expectation, $\mathbf{E}[D_i] = (n-1)p$. Since $p \geq (\ln n)^6/n$, δ lies in $[0,1]$. Direct use of (9), (10) and (11) shows that: (i) $\Pr[D_i \geq (n-1)p(1+\delta)]$ is at most

$$\exp(-(1+\rho/2)\ln n + \sqrt{\frac{((2+\rho)\ln n)^3}{36(n-1)p}}),$$

which is at most $n^{-1-\rho/4}$ for n large enough; (ii) $\Pr[D_i \leq (n-1)p(1-\delta)]$ is at most $\exp(-(1+\rho/2)\ln n) = n^{-1-\rho/2}$. Thus, the probability of there existing *some* vertex i with D_i not in the range $(n-1)p(1\pm\delta)$, is at most $n(n^{-1-\rho/4} + n^{-1-\rho/2}) \leq 2/n^{\rho/4}$ (which in turn is vanishingly small for large n). Thus, if $\mathcal{A}_1$ is the event claimed by part (a) of the statement of Theorem 2, $\mathcal{A}_1$ happens almost surely: $\Pr[\mathcal{A}_1] \geq 1 - 2/n^{\rho/4}$.

(b) We will show (b) assuming that $2 \leq L \leq (\ln n)/\ln\ln n$. (If $L = 1$, then the only solution is the complete graph; as mentioned after Theorem 2, we may assume that $L < \lceil(\ln n)/\ln\ln n\rceil$.) Define $0 < r \leq p$ by $(n-2)_{L-1} r^L = (2+\rho)\ln n$. We will prove that a random graph H' chosen from $G(n,r)$ will almost surely have diameter at most L; since "diameter at most L" is increasing in the sense defined above, and as $r \leq p$, (13) shows that this will suffice. We present some useful bounds on r. First, since L is the smallest integer for which the condition of part (b) holds, we have $(n-2)_{L-2} p^{L-1} < (2+\rho)\ln n$. So, letting $x = p/(np)^{1/L}$, the value $(n-2)_{L-1} x^L = (n-2)_{L-2} p^{L-1} \cdot ((n-L)p/(np))$ is smaller than $(2+\rho)\ln n$. So, $r \geq x$. Thus, since $L \geq 2$,

$$nr \geq (np)^{1-1/L} \geq \sqrt{np} \geq (\ln n)^3. \tag{14}$$

Second, it is easy to see that for all $n \geq 16$ and $2 \leq L \leq n-1$, $(n-2)_{L-1}(1/2)^L > (2+\rho)\ln n$. So, $r \leq 1/2$.

Recall that a simple path in a graph is a path with no repeated vertices. We will show in fact that H' will have a simple path of length *exactly* L almost surely, between all pairs of vertices. Fix any distinct vertices u, v; we will first upper-bound the probability of H' not having such a simple path between u and v by employing Janson's inequality.

Our underlying independent random variables Z_k (in the sense of Janson's inequality), are the independent random choices for inserting the various edges according to $G(n,r)$. Note that there are $(n-2)_{L-1}$ possible simple paths of length L from u to v. (For the rest of this proof of (b), "path" will mean "simple path of length L connecting u and v".) Denote each such path P by its sequence of vertices, starting with u and ending at v: say, $[u, a_1, a_2, \ldots, a_{L-1}, v]$. Since we interpret P as a *sequence*, there is a natural notion of *precedence* among the elements of the sequence: u precedes a_1, a_1 precedes $a_2, \ldots$ etc. Also, we will call $a_1, a_2, \ldots, a_{L-1}$ the *internal vertices* of P. Let $A(P)$ be P's edge-set, $\{\{u,a_1\}, \{a_1,a_2\}, \ldots, \{a_{L-2},a_{L-1}\}, \{a_{L-1},v\}\}$. Define $B(P)$ to be the event that all edges of P are in H'.

Thus, the ("bad") event that there is no simple path of length L connecting u to v, is precisely $\bigwedge_P \overline{B(P)}$; note that this is the setting of Janson's inequality, and so we can use

(12). Now, $\Pr[B(P)] = r^L$ for each P; since $L \geq 2$ and $r \leq 1/2$ as seen above, we can take $\nu = 1/4$ in the sense of Janson's inequality. Also, the quantity M is $(1-r^L)^{(n-2)_{L-1}}$, which is at most $\exp(-r^L(n-2)_{L-1}) = \exp(-(2+\rho)\ln n)$, since $1 - x \leq \exp(-x)$ for all x. Finally, $\Lambda = \sum_{(P_1,P_2)} r^{|A(P_1)\cup A(P_2)|}$, where the sum is taken over distinct P_1, P_2 such that $A(P_1) \cap A(P_2) \neq \emptyset$. ($|S|$ denotes the cardinality of a set S.) So we have by (12) that

$$\Pr[\bigwedge_P \overline{B(P)}] \leq \exp(-(2+\rho)\ln n + 2\Lambda/3). \quad (15)$$

To upper-bound Λ, we first fix a path P_1 and bound $\sum_{P_2} r^{|A(P_1)\cup A(P_2)|}$, where the sum is taken over all paths $P_2 \neq P_1$ such that $A(P_1) \cap A(P_2) \neq \emptyset$. Suppose $|A(P_1) \cap A(P_2)| = i > 0$. Since $P_2 \neq P_1$, we have $i \leq L - 1$. We upper-bound the number of such paths P_2, as follows. There are $\binom{L}{i}$ choices for the edges of P_1 that P_2 shares; fix such a choice. Next, for each of these edges, determine which of its end-points precedes the other end-point in P_2; there are at most 2^i such choices. Next, for each of the i precedent end-points (in P_2), determine their position in P_2: there are at most L^i choices. Finally, these i edges of P_2 define at least i internal vertices of P_2; the remaining internal vertices of P_2 as well as their positions in P_2, can be chosen in at most n^{L-1-i} ways. Since this process completely determines P_2, the number of paths P_2 with $|A(P_1) \cap A(P_2)| = i$, is at most $\binom{L}{i}(2L)^i n^{L-1-i}$. Since P_1 can be chosen in $(n-2)_{L-1}$ ways, we get

$$\Lambda \leq (n-2)_{L-1} \sum_{i=1}^{L-1} \binom{L}{i} (2L)^i n^{L-1-i} r^{2L-i}. \quad (16)$$

Defining $g(i) = \binom{L}{i}(2L)^i n^{L-1-i} r^{2L-i}$, we can see that $g(i+1)/g(i) \leq L^2/(nr)$ for $i \geq 1$. Bound (14) and the fact $L \leq (\ln n)/\ln\ln n$ show that $L^2/(nr) \leq ((\ln\ln n)^2 \ln n)^{-1}$. Thus, (16) shows that Λ is at most

$$(n-2)_{L-1} g(1) \sum_{i\geq 0} ((\ln\ln n)^2 \ln n)^{-i} \leq 3(n-2)_{L-1} g(1) \quad (17)$$

for n large enough. Since $(n-2)_{L-1} r^L = (2+\rho)\ln n$, Fact 1 yields $n^{L-1} r^L \leq (2+\rho)\ln n \cdot \exp(L^2/n)$, which is at most $3(2+\rho)\ln n$ for n large enough (since $L \leq (\ln n)/\ln\ln n$). Thus, $g(1) = 2r^L L^2 n^{L-2} r^{L-1}$ is at most $6(2+\rho)\ln n \cdot r^L L^2/(nr) \leq O(r^L/(\ln\ln n)^2)$, by (14) and since $L \leq (\ln n)/\ln\ln n$. So we have from (17) that $\Lambda \leq O((\ln n)/(\ln\ln n)^2)$. Substituting this into (15), the probability that u and v are at a distance more than L, is at most $n^{-2-\rho/2}$ for all large enough n. Thus, the probability of existence of *some* pair of vertices at a distance more than L from each other in H', is at most $n^2 \cdot n^{-2-\rho/2} = n^{-\rho/2}$, a negligible quantity for large n.

(c) Recall from (a) that $\Pr[\mathcal{A}_1] \geq 1 - 2/n^{\rho/4}$. Define $s = \lceil (n-1)p(1-2\delta)\rceil - 1$ and $t = \lceil (n-1)p\delta \rceil$. Let $\mathcal{B}$ be the event that H is *not* $(n-1)p(1-2\delta)$-vertex-connected, i.e., that H has a set of s vertices, the removal of which disconnects the graph. The main idea is that if $\mathcal{A}_1$ holds, then for $\mathcal{B}$ to hold, H must have disjoint sets of vertices S and T with $|S| = s$ and $t \leq |T| \leq (n-s)/2$, such that no vertex in T is connected by an edge to any vertex outside of $S \cup T$. (This is due to the following. Suppose $\mathcal{A}_1$ holds, and that there is some set S with $|S| = s$ such that if we remove S, the remaining graph is disconnected. Let T be a connected component of *smallest* size in the remaining graph; so, we must have $|T| + s + |T| \leq n$, i.e., $|T| \leq (n-s)/2$. Next, since $\mathcal{A}_1$ holds, every vertex in T has degree at least $\lceil (n-1)p(1-\delta)\rceil$. However, since each vertex v in T has no neighbor outside of $S \cup T$, v's degree can be at most $s + |T| - 1$. This means that $\lceil (n-1)p(1-\delta)\rceil \leq s + |T| - 1$; hence, $|T| \geq t$.)

Thus, letting $NE(S,T)$ be the event that there is no edge in H between T and $\overline{S \cup T}$, $\Pr[\mathcal{B} \mid \mathcal{A}_1]$ is at most $\sum_{S,T} \Pr[NE(S,T) \mid \mathcal{A}_1]$, where the sum is over all disjoint sets S, T such that $|S| = s$ and $t \leq |T| \leq (n-s)/2$. For any given S and T, $\Pr[NE(S,T) \mid \mathcal{A}_1]$ equals $\Pr[NE(S,T) \wedge \mathcal{A}_1]/\Pr[\mathcal{A}_1]$, which is at most $\Pr[NE(S,T)]/\Pr[\mathcal{A}_1]$. Now, $\Pr[NE(S,T)] = (1-p)^{|T|\cdot(n-|S\cup T|)}$. Thus, these ideas show that $\Pr[\mathcal{B} \mid \mathcal{A}_1]$ is at most

$$(\Pr[\mathcal{A}_1])^{-1} \binom{n}{s} \sum_{i=t}^{(n-s)/2} \binom{n-s}{i} (1-p)^{i(n-s-i)}. \quad (18)$$

Next, since $t \leq i \leq (n-s)/2$, we have $(1-p)^{i(n-s-i)} \leq \exp(-i(n-s-i)) \leq \exp(-t(n-s)/2)$; also, $\sum_i \binom{n-s}{i} \leq 2^{n-s}$. So since $t \to \infty$ as $n \to \infty$, the sum in (18) is at most $\exp(-t(n-s)/3)$. We also have that $\binom{n}{s} = \binom{n}{\min\{s,n-s\}}$ is at most $n^{\min\{s,n-s\}}$. So, (18) yields

$$\Pr[\mathcal{B} \mid \mathcal{A}_1] \cdot \Pr[\mathcal{A}_1] \leq n^{\min\{s,n-s\}} \exp(-t(n-s)/3). \quad (19)$$

By doing a case analysis based on whether $s \leq n/2$ or not, and keeping in mind that $t \gg \ln n$, we can check that the r.h.s. of (19) goes to zero. Finally, since

$$\begin{aligned} \Pr[\mathcal{B}] &= \Pr[\overline{\mathcal{A}_1} \wedge \mathcal{B}] + \Pr[\mathcal{B} \mid \mathcal{A}_1] \cdot \Pr[\mathcal{A}_1] \\ &\leq \Pr[\overline{\mathcal{A}_1}] + \Pr[\mathcal{B} \mid \mathcal{A}_1] \cdot \Pr[\mathcal{A}_1] \end{aligned}$$

with $\Pr[\overline{\mathcal{A}_1}]$ being vanishingly small, $\Pr[\mathcal{B}]$ tends to 0 as n increases.