Balanced Overlay Networks (BON): An Overlay Technology for Decentralized Load Balancing

Jesse S.A. Bridgewater, P. Oscar Boykin, *Member*, *IEEE*, and Vwani P. Roychowdhury

Abstract—We present a novel framework, called balanced overlay networks (BON), that provides scalable, decentralized load balancing for distributed computing using large-scale pools of heterogeneous computers. Fundamentally, BON encodes the information about each node's available computational resources in the structure of the links connecting the nodes in the network. This distributed encoding is self-organized, with each node managing its in-degree and local connectivity via random-walk sampling. Assignment of incoming jobs to nodes with the most free resources is also accomplished by sampling the nodes via short random walks. Extensive simulations show that the resulting highly dynamic and self-organized graph structure can efficiently balance computational load throughout large-scale networks. These simulations cover a wide spectrum of cases, including significant heterogeneity in available computing resources and high burstiness in incoming load. Prior analytical results show BON's scalability for truly large-scale networks; under certain ideal conditions, the network structure converges to Erdös-Rényi (ER) random graphs. Our simulation results, however, show that the algorithm does much better, and the structures seem to approach the ideal case of d-regular random graphs. We also make a connection between highly-loaded BONs and the well-known ball-bin randomized load balancing framework.

Index Terms—Distributed computing, random walks, load balancing, random graphs, randomized algorithms.

1 Introduction

Distributed computing was one of the earliest applications of computer networking, and many different methods have been developed to harness the collective resources of networked computers. Some important architectures include centralized client-server systems, distributed hash table (DHT) systems, and diffusive algorithms. Here, we introduce the concept of balanced overlay networks (BON), which take the novel approach of encoding the resource balancing algorithm into the evolution of the network's topology. Each node's in-degree is kept proportional to its unused resources by adding and removing edges when resources are freed and consumed. As we will show, this topology makes it possible to efficiently locate nodes with the most free resources, which in turn enables load balancing with no central server.

This work makes several novel contributions to distributed computing and resource sharing. First, BON is decentralized and scalable with known lower bounds on balancing performance. While other related decentralized load-balancing algorithms (e.g., Messor [1]) have been proposed in the literature, analysis of performance and scalability has been largely unavailable.

Under certain ideal conditions, we show that the network structure converges to a random graph that is at least as regular and balanced as Erdös-Rényi (ER) graphs. Second, the algorithms and protocols for both network maintenance and job allocation are based only on local information and actions: Each node decides the amount of resource or computing power it wants to share, and it embeds this information into the network structure via short random walks; similarly, jobs are distributed based only on information available through local explorations of the overlay network. Thus, BON is a truly self-organized dynamical system. Third, since the BON algorithm produces dynamic random graph topologies, these resulting networks are very resilient to random damage and also have no central point of failure. Finally, we make a connection between the performance of BON in some regimes with ball-bin random load balancing problems [2].

It is also important to note that BON is a novel paradigm for resource sharing of any kind and its applicability is not limited to distributed computing. The in-degree of a node can be made to correspond to any type of shareable resource. Then, one can exploit the fact that BON networks have low diameters associated with random graphs, which makes them easy to sample using short random walks. Extensive simulation results support the effectiveness of this approach in networks with a wide range of resource and load distributions. These simulations show that the actual performance of the algorithm far exceeds the lower bounds mentioned above and achieves very close to optimal performance.

BON is a very simple, realistic, and easily implementable algorithm using standard networking protocols. The completely decentralized nature of the algorithm makes it very well-suited to massive applications encompassing very large ensembles of nodes. The following are a few examples of applications for which BON is very well suited:

- *J.S.A. Bridgewater and V.P. Roychowdhury are with the Electrical Engineering Department, University of California, Los Angeles, 56-125B Engineering IV Building, Box 951594, Los Angeles, CA 90095-1594. E-mail: jesse.bridgewater@gmail.com, vwani@ee.ucla.edu.*
- *P.O. Boykin is with the Department of Electrical and Computer Engineering, University of Florida, PO Box 116200, 377 Larsen Hall, Gainesville, FL 32611-6200. E-mail: boykin@pobox.com.*

Manuscript received 20 Oct. 2005; revised 30 June 2006; accepted 18 Aug. 2006; published online 9 Jan 2007.
Recommended for acceptance by D. Trystram.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0442-1005.
Digital Object Identifier no. 10.1109/TPDS.2007.1031.

Single-System Image (SSI) LAN/WAN clusters: BON can be used for single-system image (SSI) clusters in the same way that Mosix [3] is used but without the need for all nodes to be aware of each other as is the case in Mosix. This can allow BON to scale to very large system sizes.

Public Resource Computing: BON is also applicable to @HOME-style projects [4]. These projects are typically special-purpose for each application. The decentralized nature of BON will allow multiple projects to share the same pool of computers.

Grid Computing: BON also has the potential to be integrated with GRID [5], [6] systems for efficient resource discovery and load distribution across virtual organizations (VOs).

Web Mirroring: Distributed Web mirroring is an example of a noncomputational application of the BON algorithm. The system could allow a huge number of software users to participate in providing download mirrors.

This paper is organized as follows: Section 2 describes prior related research in distributed computing and graph theory. Section 3 introduces the BON architecture. Section 4 discusses the theoretical analysis of BON's scalability. Section 5 provides a description of the simulation setup and results. Section 6 deals with practical considerations for implementing BON. Finally, Section 7 discusses results, conclusions, and future work.

2 Related Work

2.1 Random Graphs

A graph, $G = (V, E)$, consists of a set of vertexes, V, and a set of edges, E. Graphs provide a convenient mathematical abstraction for studying networks of nodes that are connected by links.

2.1.1 Erdös and Rényi (ER) Random Graphs

One of the most important types of random graphs, the Erdös and Rényi [7] (ER) random graph, can be defined as a set N of nodes with each of the $\frac{N(N-1)}{2}$ possible edges existing with probability p. The most surprising feature of this model is that there are distinct phase transitions in the structure of the graph as the connection probability, p, changes. The mean degree of a graph is related to the connection probability in the following way: $\langle k \rangle = p(N-1)$.

Since we are interested in using the topological properties of a graph to perform load balancing, it is important to look at a graph's connectedness properties. Fig. 1 shows that an ER graph will have a giant component if $\langle k \rangle > 1$. This means that the largest connected component in the graph has $c(p)N$ nodes and all other components are exponentially smaller with $O(\ln N)$ size. Furthermore, as $\langle k \rangle$ approaches $\ln N$, the fraction of nodes in the giant component, $c(p)$, becomes 1 with a nonzero probability.

Another important hallmark of random graphs is low diameter. Diameter is defined as the maximum of the shortest distance over all pairs of nodes. For ER graphs with $\langle k \rangle \geq 3.5$, the diameter of the graph is $O(\ln N / \ln\langle k \rangle)$. For larger average degrees, the diameter scales the same but rapidly approaches $\ln N / \ln\langle k \rangle$ when $\langle k \rangle \geq \ln N$. If we wish

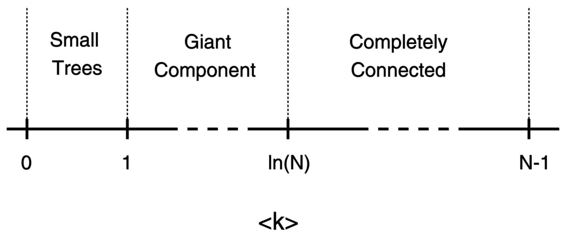

Fig. 1. The connectedness of an Erdös and Rényi random graph has several distinct phases depending on average degree. When $\langle k \rangle < 1$, the graph is a forest of trees with $O(\ln N)$ size. A giant component ($O(N)$ size) abruptly emerges as $\langle k \rangle$ exceeds 1. When the average degree exceeds $\ln N$, the graph becomes a single connected component.

to uniformly sample nodes using random walks (starting from any node in the network), it is clear that the walk length has to be at least as long as the diameter; hence, networks with short diameters are desirable.

2.1.2 Sampling Nodes via Random Walks

We have already mentioned that sampling using random walks is at the heart of BON. A random walk on any graph is equivalent to following a sample path of an associated Markov chain, where the nodes represent the states and the transition probability p_{ij} of going from node $i \rightarrow j$ is given by $p_{ij} = 1/degree(i)$ for an *undirected* graph (where each edge is basically replaced by two-directed edges pointing in opposite directions) and by $p_{ij} = 1/out\text{-}degree(i)$ for any *directed* graph; of course, $p_{ij} = 0$ if there is no edge (no directed edge, in the case of directed graphs) connecting node i to j. Thus, for any *undirected* graph, a random walk of sufficient length samples nodes according to the steady-state distribution of the underlying Markov chain,

$$\pi_k = \frac{deg(k)}{\sum_{i \in V} deg(i)}, \tag{1}$$

which is the probability of being at node k. Hence, in steady state, nodes are sampled linearly preferentially and edges are sampled uniformly. In the case of *directed* graphs, a steady state distribution exists only if the underlying Markov Chain is ergodic; a sufficient condition for ergodicity is that the network be *strongly connected*, i.e., there is a directed path from every node to every other node in the network. The steady state distribution for random walks on such networks is no longer given by the simple formula stated above; however, the nodes with high in-degrees and "centrality" [8] have larger limiting probabilities. Thus, even for directed graphs, random walks sample nodes preferentially (although, not necessarily linearly) with respect to their in-degrees; other factors, such as the in-degrees of the nodes connecting to the given node, also play a role. For example, the steady state distribution of random walks on the World Wide Web network, as defined by hyperlinks, determines the so-called "Page Rank" of a particular Web page [9].

The length of the random walk required to be very close to the steady-state distribution is referred to as the mixing time. Aldous and Fill [10] provide a detailed analysis of random walk mixing on several graph topologies including ER graphs, regular graphs, and grown preferential attachment graphs. In all of these cases, the mixing time of random walks is of logarithmic order in the size of the graph. This is important because it means that $O(\log N)$ random walks can sample the nodes preferentially with respect to degree (or in-degree, in the case of strongly connected directed graphs).

2.1.3 Regular Random Graphs

Another important model of random graphs is based on a random d-process. Given n nodes, edges are randomly added one by one with the constraint that no node has more than d incident edges. This simple model has been shown to generate random graphs where each node has the same degree (except for a single node when dn is odd) [11].

The regular graphs produced by the random d-process also have predictable connectedness [12]. For $d \geq 3$, $\lim_{n\to\infty} P(G \quad is \quad connected) = 1$. Also, d-regular random graphs have logarithmic diameter. Recent results have also shown that regular random graphs have fast mixing times.

In this paper, one of the goals is to design a network dynamic that would generate *directed* regular random graphs in the sense that each node will have the same in-degree d and an average out-degree of d. Moreover, we want these directed graphs to be strongly connected and have low diameter. We are not aware of analytical results for the minimum d (analogous to that of the undirected case) that guarantee such connectivity properties for the directed graphs. However, since $d \geq 3$ guarantees such properties for the undirected case, we enforce that our protocol maintains a minimum in-degree, $k_{min} = 4$ (hence, an average out-degree of 4, as well). Since the undirected average degree is now at least 8, we would expect that the undirected version of the graph would always be connected and that the directed graph will not have too many strongly connected components (SCCs). Indeed, our simulation results show that, when the network is in this regime, i.e., all nodes have in-degree of 4, it always remains *weakly connected* (i.e., the undirected version of the graph is connected), and has only a very small number of SCCs. As explained in detail in the simulation section, such connectivity properties seem sufficient to guarantee almost optimal load balancing in the BON system.

2.2 Distributed Computing

The authors have previously considered load balancing that is topologically based with a simpler model than BON that is amenable to analytical study [13]. In that work, each node's resources are kept proportional to its in-degree, and load is distributed by performing a short random walk and migrating load to the last node of the walk. As shown in [13], this method produces Erdös-Rényi (ER) random graphs for the uniform resource case and exhibits good load-balancing performance. However, this work lacked key components necessary to realize a robust, practical, and efficient load balancing system. For example, the algorithm aims to produce ER random graphs in the uniform-resource case and, thus, is not capable of approaching optimal load balancing performance. This is clear since optimal balancing on uniform-resource nodes would be for every node to have the same number of jobs. Furthermore, the ER formulation does not address the case where nodes become overloaded; if the degrees of the nodes are still kept proportional to their resources in the overloaded regime, then the average degree will go below the critical values required to keep the robust connectivity properties of the network, leading to deterioration in the load balancing performance. As we demonstrate in the current work, performing more complex functions on the random walk, and clipping the minimum degree, can significantly improve performance over a wide range of load conditions.

The majority of distributed computing research has focused on central server methods, DHT architectures, agent-based systems, randomized algorithms, and local diffusive techniques [1], [2], [14], [15], [16], [17], [18]. Some of the most successful systems to date [4], [19] have used a centralized approach. This can be explained by the relatively small scale of the networked systems or by special properties of the workload experienced by these systems. However, since a central server must have $O(N)$ bandwidth capacity and CPU power, systems that depend on central architectures are unscalable [20], [21]. Reliability is also a concern since a central server is a single point of failure. BON addresses both of these issues by using $O(logN)$ maximum communications scaling and no single points of failure. Furthermore, since the networks created by the BON algorithm are random graphs, they will be highly robust to random failures.

The Messor project [1], in particular, has the same goal as BON, which is to provide self-organized, distributed load balancing. The agent-based design of Messor also involves performing random walks on a network to distribute load. However, BON is designed specifically to reshape the network structure so it can be efficiently sampled. Messor was inspired by the notion of a swarm of ants that wander around the network picking up and dropping off load. Thus, it is not clear how long the ant agents will need to walk while performing the load balancing. It is the focus on topology that distinguishes BON from other similar efforts. BON endeavors to reshape the network topology to make resource discovery feasible with $O(\log N)$ length random walks. A simplified version of BON can be analyzed and, thus, we can put performance bounds on its behavior. Messor, while very interesting, provides no analytical treatment.

Within the large body of research, some techniques have been implemented including Mosix, Messor, BOINC, Condor, SWORD, Astrolabe, INS/Twine, Xenosearch [1], [3], [4], [15], [19], [22], [23], [24], and others. Many of these systems focus on providing a specific desired level of service for jobs. This contrasts to the approach taken by BON, Mosix, and others, in which processes are migrated to nodes where they will have the most resources applied to them rather than a specific level of resources. The other systems are mostly based on DHT architectures and provide for querying based on arbitrary node attributes and link qualities. For complex distributed applications where each participating node must have a certain level of resources and where the connectivity between the nodes must have prescribed latencies, these DHT systems will be the most suitable platform. For many types of parallel scientific computing, however, BON's objective of placing a

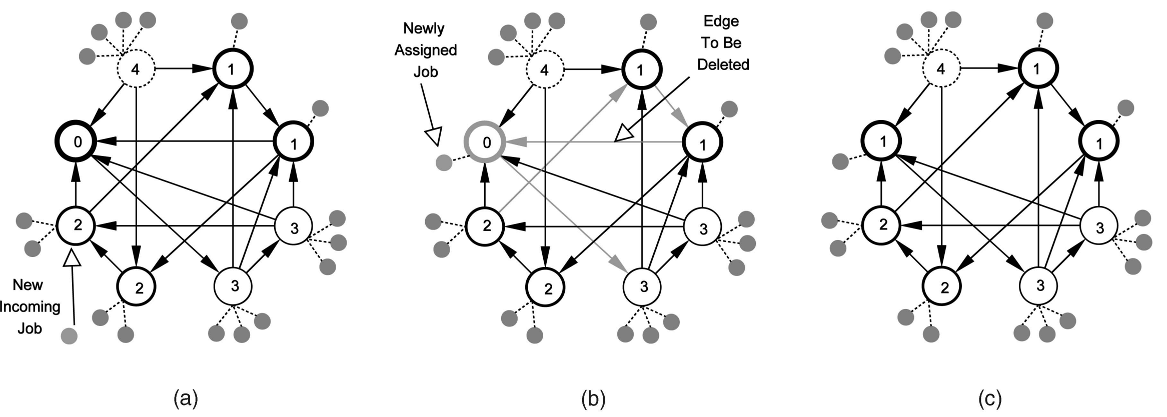

Fig. 2. New jobs are assigned by using a greedy random walk. The large nodes depict computers in a schematic BON network while the small filled nodes are jobs running on the node to which they are connected. The label for each of the computer nodes denotes the current number of jobs it is running. (a) New load enters the network. (b) We see that the node where load arrives initiates a random walk which keeps track of the degree (free resources) of each visited node. The largest degree node (most free resources) is selected to run the new load. To compensate for the additional load, the node which accepted the new load deletes one of its incoming edges to account for its diminished resources. (c) The resulting network.

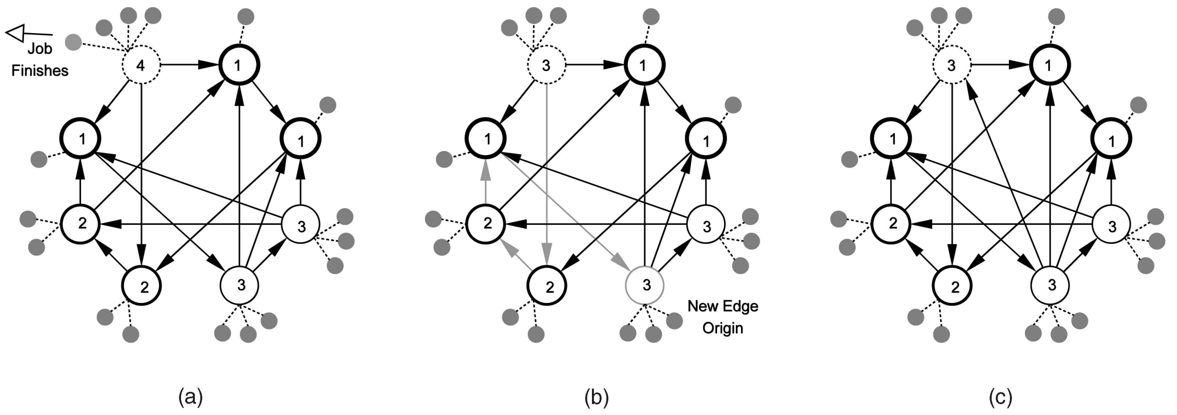

Fig. 3. When a running job finishes, the host node may need to increase its connectivity to advertise its increased resources. (a) A job finishes and, thus, leaves the network. (b) The node where the load finishes initiates a random walk. (c) The last node on the walk will be the origin of a new edge incident on the walk initiator. This new edge represents the increase in available resources on the node where the job just completed.

job where it will finish as quickly as possible is appropriate and desirable.

BON is designed to be deployed on extremely large ensembles of nodes. This is a major similarity with BOINC [4]. The Einstein@home project, which processes gravitation data, and Predictor@home, which studies protein-related disease, are based on BOINC, the latest infrastructure for creating public-resource computing projects. Such projects are single-purpose and are designed to handle massive, embarrassingly parallel problems with tens or hundreds of thousands of nodes. BON should scale to networks of this size and beyond while providing a dynamic, multiuser environment instead of the special-purpose environment provided by BOINC.

3 The BON Architecture

3.1 BON Topology

The concept underlying BON is that the load characteristics of a distributed computing system can be encoded in the topology of the graph that connects the computational nodes.

In schematic terms, an edge in a BON graph represents a certain unit of unused capacity on the node to which the edge points. Consequently, when a node's resources are being exhausted, its in-degree will decline, as seen in Fig. 2. Conversely when a node's available resources are increasing, its in-degree will rise, as seen in Fig. 3.

Formally, a BON is a dynamic, directed graph, $D = (E, V)$, where each node $v_i \in V$ maintains $k^{(min)} \leq k_i(t) \leq k_i^{(max)}$ incoming edges. The maximum incoming edges that a node can have, $k_i^{(max)}$, is proportional to the computational power of v_i; the role of the minimum in-degree is described in the following: Each node v_i has a scalar metric, $s_i(t)$, which is kept inversely proportional to $k_i(t)$. As $s_i(t)$ changes with time, v_i severs or acquires new incoming links to maintain the relationship between degree and free resources (Fig. 4). In the context of distributed computing, $s_i(t)$ is a scalar representation of the current load experienced by node v_i. This means that each node will endeavor to keep its in-degree proportional to its free resources or inversely proportional to its load. Idle nodes will have a relatively large in-degree while overloaded nodes will have a small in-degree. The total unloaded resources of a node are proportional to its maximum in-degree, $k_i^{(max)}$. As long as the network is not overloaded (i.e., no node is forced to clip its in-degree to the minimum of $k^{(min)}$), each node's free resources should be proportional to its degree, and **a uniform-resource network is ideally balanced if the graph structure is d-regular**.

Note that, as the overall network load increases, the average in-degree of the nodes decreases, forcing the nodes to clip their in-degrees to k^{min}, and the free-resources versus in-degree correlation is no longer kept valid. We demonstrate, however, that, even in this overloaded regime, we achieve almost-optimal load balancing and the network topology approaches the structure of a directed regular random graph. Recalling the properties of regular random graphs, selecting $k^{(min)} = 4$ ensures that the undirected view of a BON will be connected and have a low-diameter.

3.2 BON Algorithm

Each node's load, $s_i(t)$, can change as a new load arrives in the network or when existing work is done. When a new load arrives at v_i, a short random walk is initiated to locate a suitable execution site. The nodes visited by the random walk will have degree that is proportional to its free resources. Contained in this random walk is a BON resource discovery message (BRDM) that stores the merit function information for the most capable node visited so far on the walk.

Due to the mapping between load and in-degree, this greedy random walk selection is the the same as choosing the highest degree node encountered in the walk, as long as no node in the random walk is overloaded (i.e., its free resource maps to an in-degree that is less than the allowed value of $k^{(min)}$), forcing it to clip its in-degree to $k^{(min)}$. Thus, the BON algorithm has two primary regimes. First, when $\langle k \rangle \gg k^{(min)}$, the BON algorithm is in the ***linear regime***, i.e., the free resources of nodes are linearly proportional to their in-degrees. In this linear regime, the random walk samples the nodes with a frequency that is preferential in the in-degree of the nodes (see Section 2); in fact, our simulations show that the sampling frequency is indeed proportional to the in-degree. Hence, nodes are sampled preferentially with respect to their free resources in the linear regime. When $\langle k \rangle = k^{(min)}$, BON is in the ***clipped regime***. Thus, in the *clipped regime*, the free resources of the nodes are no longer encoded by their in-degrees; in fact, we find that all nodes have the minimum allowable in-degree. Hence, nodes are being sampled randomly and not preferentially with respect to their free resources in the load-clipped regime. In practice a node in the clipped regime will be under very heavy computational load, but, fundamentally, it can still accept new jobs. We will discuss the case when the network is load clipped in Section 5.4, and show that load balancing is still performed efficiently in this regime.

Another approach to choosing a node on the walk is to select the last node inserted into the BRDM. This case has been previously explored [13] in the linear regime. While this simple approach can be studied analytically, simulation results indicate that large improvements to the balancing performance are possible by always keeping the most capable node's information. Instead of performing a simple random walk and selecting the last node to receive incoming load, the node on the walk with the largest power per load will be the target. This choice is made because it selects the node that has the most free resources to offer to the next incoming job. This same prior work also showed that starting a BON in an ordered ring or in a random configuration both yielded the same steady state random graphs.

4 Analytical Background

The performance of greedy BON walk selection is lower-bounded by the performance of the standard walk selection considered in [13]. Therefore, although we do not present a calculation of the load distribution for BON graphs, we can state that it has the same scalability as the standard walk case described below. We also calculate the bandwidth used by the algorithm and compare it to a centralized model.

4.1 Scalability

4.1.1 Linear Regime

Recall that, in the linear regime (i.e., when the average degree, $\langle k \rangle > k^{(min)}$), the in-degrees of the nodes are kept linearly proportional to their free resources. Since random walks in steady state will sample nodes preferentially with respect to the in-degree of the nodes (see Section 2), a random walk sampling of sufficient length on the BON network, operating in the linear regime, corresponds to sampling nodes preferentially with respect to their free resources. The greedy BON algorithm is difficult to study analytically due to the fact that we pick the node with the maximum free resources, among the preferentially sampled nodes, as our load destination node. However, prior results [13] show that a modified BON, where a preferentially sampled node is picked as the destination, is more amenable to analysis.

Rather than selecting the node on the BRDM walk that can process an incoming job the fastest, one can simply select the last node of the walk. Such a node represents a preferentially (with respect to its in-degree) or equivalently (with respect to its free resources) sampled node. As discussed in Section 2, in a directed network, this preferential selection is not necessarily linear; however, our extensive simulations show that a linear approximation is highly justified for the BON networks created by our dynamic protocols and, in this section, we will assume a linear preferential sampling. In this model, the average number of absent edges, J, in the N-node graph is identified as the total number of jobs running. The maximum number of incoming edges that a node can have will be called C and the number of incoming edges to a given node is denoted as i. For the case when the average number of jobs remains constant, we can describe this system as a simple Markov process with state-dependent arrival and service rates; it can be denoted by the standard queuing notation as $M/M/\infty//M$. The arrival rate of new jobs is proportional to the free resources, $i/(NC - J)$, of each node since jobs arrive preferentially based on in-degree. Assuming that jobs terminate uniformly randomly, the departure rate is $(C - i)/J$. Solving the birth and death Markov process, we obtain for the degree distribution

$$p_n = \binom{C}{n} \left(1 - \frac{J}{NC}\right)^n \left(\frac{J}{NC}\right)^{C-n}. \qquad (2)$$

Defining the normalized load as $\alpha = J/NC$, the binomial distribution means that, for each node, each unit of capacity

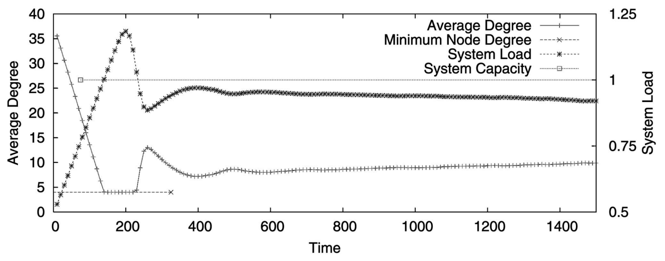

Fig. 4. The relationship between load and node degree is the basis for the BON algorithm; a node with a high in-degree is more likely to be visited on a random walk and is thus more likely to be the recipient of new load than a node with a low in-degree. Each node is defined to have P units of resources. The load is the number of running jobs, J, divided by P. As the total load increases, $\langle k \rangle$ decreases until the load becomes clipped. In the load clipped regime, the algorithm remains the same but the mechanism behind the performance changes to become a ball-bin load balancing problem with $\log N$ choices. This change is due to the fact that there is no longer a connection between free resources and in-degree. The parameters for this 2048-node simulation are given in Table 1.

is occupied with probability α. If $C = N - 1$, this model recovers the degree distribution for ER graphs:

$$p_n = \binom{N-1}{n}\left(\frac{\mathrm{E}}{\mathrm{N(N-1)}}\right)^{\mathrm{n}}\left(1-\frac{\mathrm{E}}{\mathrm{N(N-1)}}\right)^{\mathrm{C-n}}, \quad (3)$$

where $E = N(N-1) - J$.

For a network with uniform resource distribution, the variance of the degree distribution maps directly onto the variance of the load. This is because each incoming edge represents a quantum of unused power. Here, the load of each node is defined as $\frac{jobs}{power}$. In a perfectly balanced network, each node will have the same free resources. This ideally balanced network is a regular graph and, thus, the variance of the degree distribution and the load distribution is zero. This contrasts with the ER case mentioned above, which has a binomial variance. The highest-degree (most free resources) node visited on a random walk must have greater or equal resources than the last node on the walk.

In addition to this queueing model, it has been shown by information theoretic arguments [13] and simulations that the simplified rewiring protocol described here creates ER random graphs.

4.1.2 Clipped Regime

Recall that in the clipped regime, the average load is high enough that individual nodes can no longer keep its in-degrees proportional to their free resources, and the protocol forces them to clip their in-degrees to $k^{(min)}$. As demonstrated in later sections, BON produces regular directed random graphs with an in-degree of $k^{(min)}$. In Section 2.1.3, we reviewed the result that random d-regular undirected graphs with $d \geq 3$ are connected and fast mixing. We find, via extensive simulations, that the same results hold for the directed regular BON networks that we create by choosing $k^{(min)} = 4$; thus, BON networks of any size can be randomly sampled with short random walks. Therefore, BON random walks of length K essentially perform K random selections from the network. Then, selecting the least-loaded node from the K selections takes advantage of the power of K choices [2]. Our extensive simulations support this analysis, and we find that the BON system can provide effective load balancing in the clipped regime as well.

TABLE 1
Parameters for Uniform/Poisson Simulations

Uniform/Poisson Simulations			
Figures	Node Degree k	Poisson Job Size Distribution ν	Job Arrival Rate b
5, 8, 10,	71	400	512
4	71	268	512
7	71	266	512

In all of these cases, $N = 2{,}048$. Here, we see which simulations correspond to each of the figures and we give the degree distribution, job size distribution, and arrival rate.

4.2 Communications Complexity

An important metric of performance for distributed computing is the network bandwidth required for a protocol. It is clear that the architecture that requires the least total bandwidth is a central server. While the total consumption of bandwidth is important, the bandwidth that any single node consumes can be a significant bottleneck for large central networks. Below, we compare the bandwidth required by a centralized algorithm and by BON, and find that BON provides significant reduction in the maximum bandwidth consumed by any node in the network.

4.2.1 Centralized

The simplest nontrivial centralized architecture for a computing network is the case where, initially, the central node, denoted LB, knows the power and load of each of the N nodes that it controls. When a job on one of the nodes completes, that node will notify LB so that it can update its load state information for the network. Obviously, LB keeps track of the assignments of new load to each of the nodes. This method does away with the need to periodically probe every node in the network; however, it is clear that the bandwidth, memory, and CPU cost that LB has to bear is still $O(N)$. Further, assume a steady-state network load and that, in every time unit, $N\beta$, jobs begin and the same number terminate. We further assume that all the jobs will start at one of the computational nodes and that they will then be sent to LB for assignment. Now, assume that, for every job that is started, a relatively large A-byte packet, including the size of the program code and input data, must be sent from LB to each of the nodes, N_i, $i \in \{1, \cdots, N\}$, and that relatively small a-byte packets must be sent to the central server in response to changes in load. Therefore, LB must send $N\beta A$ bytes per unit time which consumes kernel resources and requires bandwidth that increases with N. The total bandwidth consumed by the entire centralized network is

$$B_T^{(LB)} = N\beta(A + a). \quad (4)$$

This is also the same amount that LB must consume since it is involved in every communications round. For all nodes $N_i, \forall i \in \{1, \cdots, N\}$, the bandwidth consumed will be $B_T^{(i)} = \beta(A + a)$, which is $O(1)$.

4.2.2 BON

For the decentralized BON algorithm, the network topology is now more complex than for the central server. While the graph of the central model was a star, BON will look approximately like a random regular graph. Initially, we will assume that we begin with a correctly formed BON. As with the central model, we assume that $N\beta$ jobs begin and end at random nodes in each time unit. Since there is not a central server, each node that initiates a new job must send a BRDM to find a node to run the job. Every node on the walk will need to replace the value of $obj = \frac{power}{load+1}$ (L bytes of data) in the BRDM if obj is larger than the objective function that is currently in the BRDM. Since the random walk will be $O(\log N)$ steps long, the total bandwidth of the walk will be $B_w = a \log N$. Likewise, when a job finishes, another walk will be used to find a replacement for the removed edge. Factoring in the cost of transmitting the program to the target node and needed handshaking protocols, the total bandwidth consumed by BON is

$$B_T^{(BON)} = N\beta(A + a(\log N + 2)). \tag{5}$$

Therefore, the total bandwidth cost of BON is $O(\log N)$ greater than the central model, i.e., $O(N)$ versus $O(N \log N)$. As already mentioned, a more important metric in many situations is the maximum bandwidth consumed by any of the nodes. In BON, each node will consume bandwidth in proportion to how many jobs it initiates and how powerful it is. Thus, if all of the nodes use the network equally, then each node will consume $B_T^{(BON)}/N$ bandwidth, which scales as $O(\log N)$. This contrasts to the $O(N)$ bandwidth needs of the central server.

5 Simulations

5.1 Simulation Description

For the simulations, each node in a BON network is a computer with power equal to its maximum degree minus its minimum degree, $P_i = k_i^{(max)} - k^{(min)}$. One unit of power can process a unit of load in each unit of time. Jobs run on these computers in a time-sharing fashion with each of the L jobs of a computer equally sharing the node's power at each time step. The simulations deal only with CPU power as the objective function of the balancing. Other features such as memory ushering will not be simulated but will be added as features in the reference implementation. Simulations of the BON system were performed using the Netmodeler [25] package. Two types of experiments were done.

The first experiments (Table 2) are very idealized, using uniform node power, uniform job arrival rates, and Poisson-distributed job sizes. This setup may apply to cluster computing or other systems where hardware is homogeneous and load is predictable.

The second type of simulation (Table 2) uses power-law distributions for all parameters including job arrival rate, node power, and job size. This configuration represents a situation where every important system parameter is distributed in a bursty, heavy-tailed way. Heavy-tailed distributions are common in many real systems [26], including networks, and, thus, these simulations provide a fairly realistic idea of how the system will perform under real loads. Most importantly, for simulation performance, the computing power ranges from 1 unit of power to 300 units of power. This is at least 10 times the range of performance seen in commonly used CPUs. As we will revisit in the performance evaluation, having many nodes that can only accept a few processes prior to being load-clipped will impact the balance distribution simply due to quantization effects. This issue will have design implications for the implementation.

TABLE 2
Parameters for Power-Law Simulations

Power-Law Simulations						
Figures	Degree Distribution $P(k) \sim k^{-1}$		Job Size Distribution $P(j) \sim j^{-1}$		Job Batch Size Distribution $P(b) \sim b^{-1}$	
	k_{min}	k_{max}	j_{min}	j_{max}	b_{min}	b_{max}
9	5	304	32	1024	1	4250
6, 11, 12, 13	5	304	32	1024	1	6000

In all of these cases, $N = 2{,}048$. Here, we see which simulations correspond to each of the figures. The degree distribution, job-size distribution, and job arrival batch size are all power-laws with the given minimum and maximum values.

In all of these simulations, we begin with a randomly connected network subject to the initial degree distribution. However, if one starts with a completely ordered ring with $O(N)$ diameter, the graph will quickly converge to the low-diameter structure depicted in these simulations. This convergence was previously demonstrated [13]. Since graph structure is so important to this algorithm, *all jobs are initiated on a single node* rather than allowing jobs to arrive at random nodes. This choice means that the balancing of load is due to the random walk distribution and not to random placement.

5.2 Graph Structure

The idea at the heart of BON is that the graph structure can capture the load state of a computational network. In Section 4, we discussed prior theory results that describe the structure of graphs formed using algorithms similar to BON. We now present simulation results for both the uniform and heavy-tailed systems described above. The degree distribution for a balanced overlay network matches the resources of the constituent nodes for both uniform and power-law resource distributions as seen in Figs. 8 and 13.

Figs. 5 and 6 show that BONs maintain a low diameter and exhibit the property of random graphs that the diameter is proportional to $\ln N / \ln\langle k\rangle$. The changes in connectivity can be seen in Fig. 7.

It is important that BON graphs remain at least weakly connected (i.e., the undirected version of the network is connected or has a single connected component) as they evolve. All simulations indicate that BONs remain weakly connected but that, when they are load-clipped, they can acquire multiple strongly connected components (SCCs). Recall that an SCC is a set of nodes where there is at least one directed path from each node to every other node in the

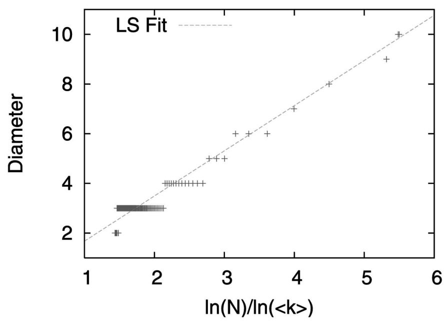

Fig. 5. BON graphs obey the random graph scaling relationship between diameter (undirected) and average degree: $Diameter \propto \ln N / \ln\langle k\rangle$, where N is the number of nodes and $\langle k\rangle$ is the mean degree. This BON graph was generated using the uniform simulation parameters from Table 1.

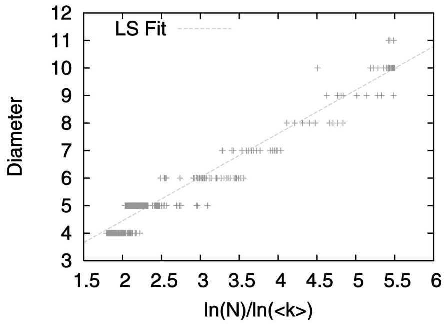

Fig. 6. The random-graph relationship between diameter (undirected) and average degree: $Diameter \propto \ln N / \ln\langle k\rangle$ also holds for the BON networks generated using the power-law simulation parameters from Table 2.

set. If there is only one SCC, then the network itself is strongly connected. The random d-process model of a random graph indicates that random regular graphs stay connected if $d \geq 3$ [12]. This result holds as $N \rightarrow \infty$. Thus, it is not surprising that nearly regular BON graphs with a minimum degree of 4 also remain weakly connected.

As the load surpasses 1 (the clipping threshold) and the network becomes a $k^{(min)}$-regular graph, the number of SCCs increases. As shown in Fig. 9, the number of SCCs falls back to unity when an overloaded network becomes less loaded. It is also important to note that the SCCs in an overloaded BON can change due to the rewiring of the network. So, while every node will not be able to communicate with every other node at each instant of time, the out-component of each node in the graph can change with time, allowing the network to recover effectively. Also, the network does remain weakly connected even when the network has many SCCs.

If future studies indicate that BON graphs are becoming disconnected, there are several measures that can be taken. First, the minimum degree can be increased. Second, each node could keep a history of all nodes that it has interacted with and use that list to attempt to reconnect if it finds itself in a small-size subnetwork.

5.3 Load Balancing Performance

When discussing load balancing performance, we want metrics that measure how closely load follows capacity. When all the nodes are equally capable, standard deviation is a convenient measure of balancing; when nodes are heterogeneous, correlation coefficient is what we use.

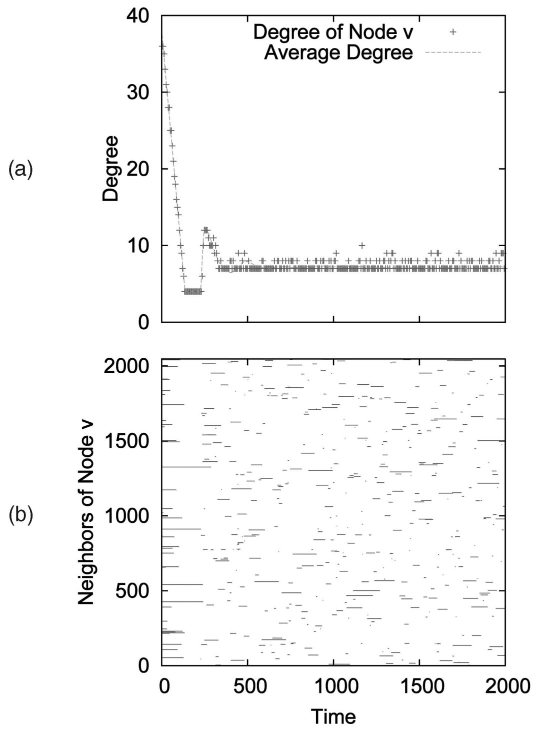

Fig. 7. As a BON network evolves, there is significant turnover in connections. (b) The incident edges of a randomly-chosen node are depicted as a function of time. Each point in this graph represents an incoming edge from the corresponding node at a certain time. (a) shows the average degree of the graph, as well as the instantaneous degree of the chosen node, whose in-coming edges are being tracked. After a short time, the graph settles down to a steady average degree, but the detailed connectivity structure changes as the graph evolves. This illustrates that the structure of the graph changes significantly even when the macroscopic properties such as average degree are not changing. The simulations parameters for this figure are shown in Table 2.

5.3.1 Simple Idealized System

In simulations of BON with uniform parameters, the load metric is the total number of jobs divided by the total capacity. For the uniform simulation model, Fig. 10 shows that the ensemble standard deviation of the node load is just below 1 percent when the network is in the underloaded regime. When the network is clipped, the standard deviation of the load is ≈ 20 percent higher than in the underloaded regime. This difference in performance is likely due to the transition from the degree-correlated load-balancing that is in effect when the network is underloaded to the ball-bin load balancing that takes over when the network is clipped.

Another important measure of performance is how well BON performs in comparison to a central system that places new jobs at the least loaded node in the network. In the uniform configuration after 1,000 iterations, the central system has completed 501,314 jobs compared with 501,238 jobs being completed by BON. This indicates that BON's job throughput is only about 0.01 percent worse than the optimal schedule.

5.3.2 Power-Law System

The power-law simulations illustrate an important design criterion for practical implementations. For these simulations, the power distribution of the nodes is a power-law

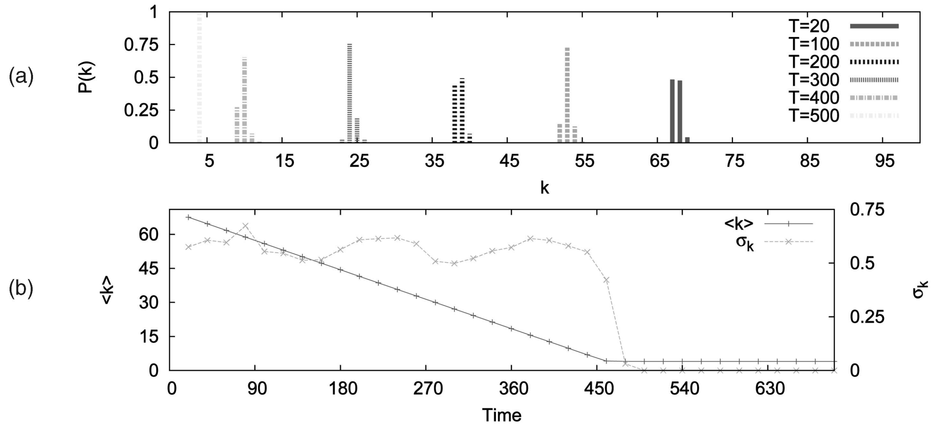

Fig. 8. BONs with uniform resources are approximately regular graphs. (a) These degree distribution snapshots are very close to delta functions. (b) The average degree, $\langle k \rangle$, and the degree standard deviation, σ_k. The standard deviation is very small, which further indicates the nearly optimal balancing.

given in Table 2. The minimum power is 1 and the maximum power is 300. Therefore, there are many nodes that have very low power resources. This means that, for many values of the load, it will be impossible to get close to optimal balancing. For this reason, the correlation between degree and free resources is used to evaluate performance as shown in Fig. 11. A good example is a node with $P = 2$. Because the load is defined to be P/L, where L is the number of running jobs, the load is limited to be nonnegative integer multiples of 1/2. Thus, if the network is 75 percent loaded, then this low-powered node is equally unbalanced whether one or two jobs are running. By selecting a suitable minimum power, one can bound this finite size effect. For example, if the least powerful node has $P = 5$, then it can always get within 10 percent of the optimal value. This finite size effect appears as cyclical behavior of the load standard deviation and can clearly be seen in Fig. 12.

As was done for the uniform simulation configuration, we compare centrally scheduled job throughput to BON throughput with the same load trace. In the heavy-tailed configuration, after 1,000 iterations the central system has completed 585,872 jobs compared with 585,788 jobs being completed by BON. As with the uniform configuration, BON's job throughput in the heavy-tailed configuration is only about 0.01 percent worse than the optimal schedule.

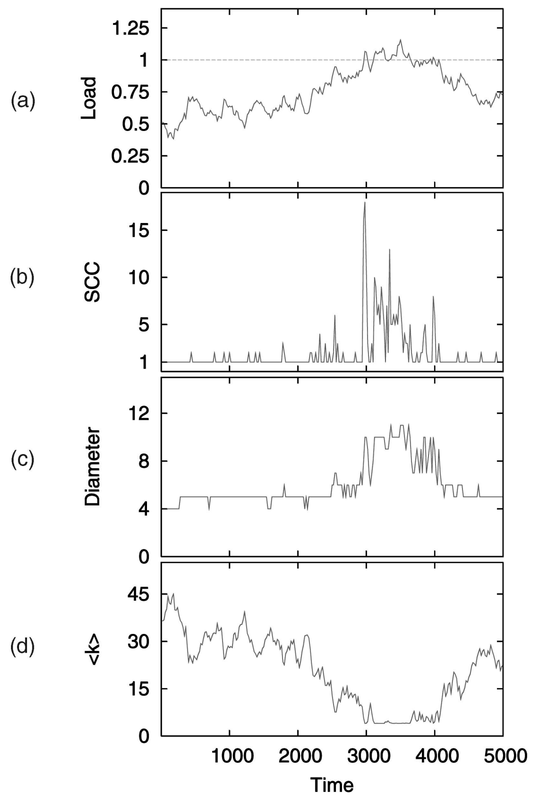

Fig. 9. BONs remain weakly connected. In simulation, we observe that BONs are always at least weakly connected directed random graphs. This 2,048-node BON with heavy-tailed parameters remains weakly connected even in the clipped regime. The number of strongly-connected components (SCC) does increase under heavy load, but the number of SCCs returns to unity when the clipping condition passes.

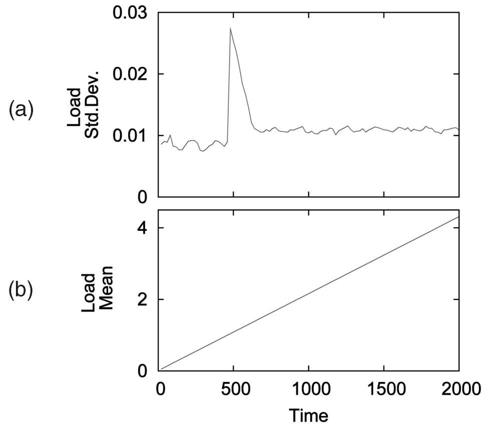

Fig. 10. This uniform resource, 2,048-node BON under increasing overload also shows that the standard deviation of the load is low at about 1 percent. The difference in performance as the network enters the clipped regime can also be seen. At the clipping transition point, the standard deviation experiences a spike, which is likely due to jobs accumulating in a small SCC before additional rewiring can balance the load. After a short time, this load imbalance dissipates as the rewiring allows a load to be distributed throughout the network, although with a standard deviation that is about 20 percent higher. This reduction in performance is not surprising, given that sampling is no longer preferential in the clipped regime.

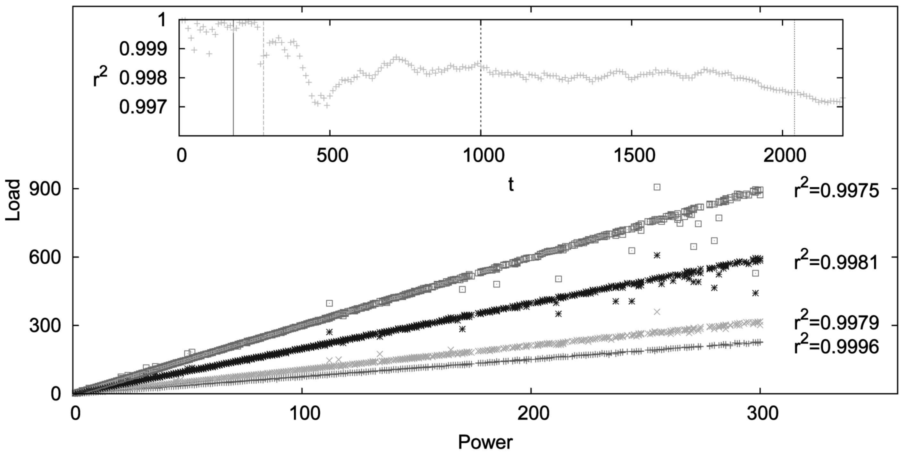

Fig. 11. A node's load is proportional to its resources. For the case where we do have many nodes with a low maximum degree, the correlation between node power in-degree is an appropriate measure of performance. For a network that is getting increasingly loaded, the load versus power is plotted at four time instants and the correlation coefficient is also calculated. Even when the network is 3 times the load clipping threshold, the correlation coefficient $r^2 > 0.99$.

Please note that this result ignores the effects of job distribution latency on total throughput, but it indicates that job placement is very close to optimal when communications delays are ignored.

5.4 Ball-Bin Regime

Every node in the graph must maintain a minimum degree to ensure that the graph stays at least weakly-connected. For these experiments, each node maintains at least four incoming edges, which means that if the network's load becomes clipped, there is no longer a correlation between a node's degree and its resources. This minimum degree was motivated by graph theory results discussed in Section 2. For this reason, the real metric that is sampled on the walk is the amount of computing power that the next incoming process can expect on a given node. When the network is not clipped, this is the same as choosing the highest-degree node on the walk. However, for a clipped network, it selects the node on the walk that has the largest value of the expected power for the next incoming job. Now, consider that a clipped network is approximately a regular random graph and thus a short random walk will sample uniformly from the nodes in the network. This problem now shows itself to be very similar to ball-bin load balancing [2], [27]:

$$\begin{cases} \langle k \rangle > k^{(min)}, & \text{preferential sampling} \\ \langle k \rangle = k^{(min)}, & \text{ball-bin sampling.} \end{cases} \tag{6}$$

In ball-bin systems, a ball is uniformly randomly assigned to one of N bins. As this process is repeated, a distribution of bin population emerges and has been studied extensively under many kinds of assumptions. The important result from ball-bin systems is that if one probes the population of more than one bin prior to assigning a ball, the population of the most full bin will be reduced exponentially in N. This work is often referred to as the "power of two choices" [2].

In the load-clipped regime, we have a similar situation where, instead of two choices, we have the power of $\log N$ choices. Each random walk on the $k^{(min)}$-regular graph will sample uniformly randomly from the nodes. Then, the least loaded nodes from the $\log N$ choices will be the target to accept the new load. This connection is made to give intuition for why BON should function in the overloaded regime, but we will not examine this aspect of the system in detail here. Detailed follow-up work will be performed to compare overloaded BON performance to the theoretical predictions of ball-bin systems.

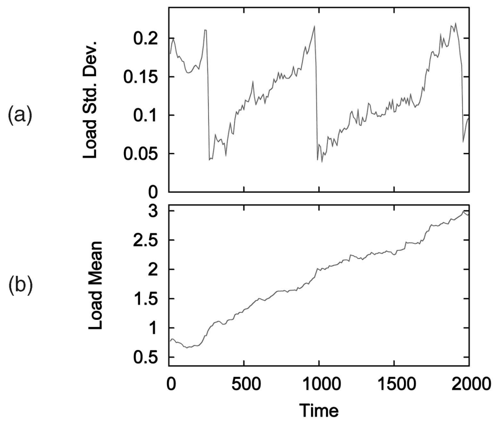

Fig. 12. Load encoding quantization. Here, we see the cyclical nature of the load standard deviation as a function of network load. The dips happen when the network is at an integer multiple of the load clipping threshold. This is due to the assumption in this model that most of the nodes have a very low degree (power-law distribution). For instance if the network is x percent loaded, a node with a maximum in-degree of $k_{min} + k_{load}$ will only be able to have the same load as the rest of the network if $\frac{J}{k_{load}} = l_{load}$, for all $J \in \{0, \cdots, k_{load}o\}$, $o \in \mathbb{Z}^+$. This makes selecting a sufficiently large minimum degree important if resources are power-law distributed.

6 Practical Considerations

6.1 Network

One strength of BON is that the data structure used on the load balancing algorithm is automatically maintained by each node through the edges that it keeps with the other nodes in the network. This means that keeping the algorithm in the correct state is the same as keeping the

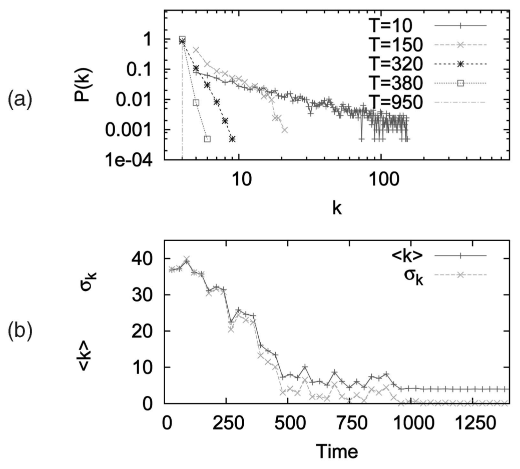

Fig. 13. For a network with a power-law resource distribution, the degree distribution, $P(k)$, also has a power-law form with the exponent getting more negative with increasing load. Here, the mean degree is $\langle k\rangle$ and the degree standard deviation is σ_k.

network in the correct state. Fortunately, this state maintenance is easily achieved using the local edge wiring rules. In the presented simulations, some networks have nodes with hundreds of incoming edges. While modern computers can easily maintain hundreds of simultaneous TCP connections, for BON, it may be preferable to use UDP for some aspects of the network.

BON nodes will interact with edges when load is distributed using random walks and when edges are being created or destroyed. These edges are important because they maintain the state of the network, but if a connection goes down, it can easily be replaced without affecting system performance.

At any step in the random walk, the algorithm does not depend on having particular connections available. If an edge is not available to be traversed due to failure, another node can be used just as well without loss of algorithmic correctness. Of course, reliable transmission is important because losing the packet will prevent a job from being migrated. There are numerous ways to provide for reliable communications between nodes using both TCP and UDP. Efforts to use fast lightweight protocols while minimizing latency will be important design issues for a BON implementation. Most connections at any given time will not be transmitting BRDMs but will be maintaining the network state. For state maintenance, the use of UDP will drastically reduce overhead compared to TCP and will allow a much larger number of edges to be maintained with less overhead than TCP. Using soft state information from packet traffic to perform keep-alive operations will help mitigate connection maintenance overhead.

6.2 State Encoding

For the load objective function, we will follow a similar approach to the Mosix SSI cluster computing system [3]. The Mosix migration algorithm is heuristic in nature and basically attempts to run processes where they will finish the most quickly. Various historical data about the process execution and node load and resources are used to judge which node can process a job with the least cost. Additionally, Mosix uses a memory ushering protocol to migrate processes away from nodes with depleted memory resources. This ushering is done in favor of trying to integrate the memory and CPU metrics into a single scalar value. These methods have been motivated by real system profiling and have proved to be successful. Thus, the node resource that will be kept proportional to in-degree is the available CPU resources of the node. In particular, we wish for new load to be assigned to the node v_i, where

$$i = argmax_j \left\{ \frac{P_j}{L_j + 1} \right\}.$$

Here, P_j is v_j's power, which can be any standardized way of representing the number of operations per unit time that a node can perform, and L_j is the number of processes competing for P_j (UNIX load). The details of how to weight integer, floating-point, and other processing characteristics will not be considered here, but it will be assumed that a reasonable benchmark of CPU performance can be constructed and run periodically on each BON node.

6.3 Load Quantization

Since computing power is represented by the edges in the network, it is important to scale the power that each edge represents in order to get the most load balancing performance for the least bandwidth and state maintenance. The initial implementation of BON will specify a CPU model to be the baseline of computational power. As computer performance changes, adaptive baselining can be performed to automatically scale how much computing power is represented by a BON edge. All other node powers are computed with regard to the $k\mathrm{th}$ percentile of benchmarks. That is, all nodes in the $k\mathrm{th}$ percentile will have the baseline power of P_b and will maintain at most $k^{(b)} - k_{min} = 5$ baseline resource edges. All other nodes will maintain

$$k_i - k_{min} = \frac{P_i(t)}{P_b} k^{(b)} \tag{7}$$

resource edges. Choosing $k_i^{(max)} - k_{min} \geq 5$, $\forall i$ ensures that even the least powerful nodes in the network can have a load that is within 10 percent of optimally balanced.

7 Concluding Remarks

Balanced overlay networks (BON) is a novel decentralized load-balancing approach that encodes the balancing algorithm in the evolving structure of the graph that connects the resource-bearing nodes. BON is scalable, self-organized, and relies only on local information to make job assignment decisions. New jobs are assigned to a node by a random walk on the graph which not only samples the graph preferentially but also selects the highest-degree node that was visited on the walk. Each node's unused resources are proportional to its degree, so this approach works very well when a network is not loaded beyond its clipping point. When a BON is clipped, the relationship between load and in-degree breaks down, but the balancing performance remains quite good due to the so-called "power of two choices" in ball-bin load balancing. Based on previous theoretical results and extensive simulation results, BON is seen to be efficient and practical. Further ongoing work on this problem includes geographical awareness extensions using more complex walk objective functions, a reference implementation on PlanetLab, theoretical analysis of the random walk with greedy node selection, algorithmic optimizations, and a full comparison of overloaded regime results with the predictions of ball-bin random load-balancing. Finally, it should be noted that this is only one possible way to encode information about a network in its topology; other distributed algorithms may benefit from using graph state to bias node selection.

REFERENCES

[1] A. Montresor, H. Meling, and O. Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents," citeseer.ist.psu.edu/montresor02messor.html, 2002.

[2] M. Mitzenmacher, "The Power of Two Choices in Randomized Load Balancing," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 10, pp. 1094-1104, Oct. 2001.

[3] A. Barak and O. La'adan, "The MOSIX Multicomputer Operating System for High Performance Cluster Computing," *Future Generation Computer Systems,* vol. 13, nos. 4-5, pp. 361-372, citeseer.ist.psu.edu/barak98mosix.html, 1998.

[4] D.P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," *Proc. Fifth Int'l Workshop Grid Computing (GRID '04),* pp. 4-10, Nov. 2004.

[5] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Int'l J. High Performance Computingr Applications,* vol. 15, no. 3, pp. 200-222, 2001.

[6] S. Adabala et al., "From Virtualized Resources to Virtual Computing Grids: The In-VIGO System," *Future Generation Computer Systems,* vol. 21, no. 6, pp. 896-909, June 2005.

[7] P. Erdös and A. Rényi, "On Random Graphs I," *Publicationes Mathematicae,* vol. 6, pp. 290-297, 1959.

[8] J.S. Kong, P.O. Boykin, B. Rezaei, N. Sarshar, and V. Roychowdhury, "Let Your Cyberalter Ego Share Information and Manage Spam," preprint, http://xxx.lanl.gov/abs/physics/0504026, 2005.

[9] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems,* vol. 30, nos. 1-7, pp. 107-117, 1998.

[10] D.J. Aldous and J.A. Fill, *Reversible Markov Chains and Random Walks on Graphs,* book in preparation, http://www.stat.berkeley.edu/~aldous/book.html, 2006.

[11] A. Rucinski and N. Wormald, "Random Graph Processes with Degree Restrictions," *Combinatorics, Probability, and Computing,* vol. 1, pp. 169-180, 1992.

[12] A. Rucinski and N. Wormald, "Connectedness of Graphs Generated by a Random D-Process," *Australian J. Math.,* vol. 72, pp. 67-85, 2002.

[13] J.S.A. Bridgewater, P.O. Boykin, and V.P. Roychowdhury, "Statistical Mechanical Load Balancer for the Web," *Physical Rev. E,* vol. 71, p. 46133, DOI: 10.1103/PhysRevE.71.046133, 2005.

[14] M.M. Theimer and K.A. Lantz, "Finding Idle Machines in a Workstation-Based Distributed System," *IEEE Trans. Software Eng.,* vol. 15, no. 11, pp. 1444-1458, 1989.

[15] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Scalable Wide-Area Resource Discovery," citeseer.ist.psu.edu/oppenheimer04scalable.html, 2004.

[16] R. Subramanian and I.D. Scherson, "An Analysis of Diffusive Load-Balancing," *Proc. Sixth Ann. ACM Symp. Parallel Algorithms and Architectures,* pp. 220-225, 1994.

[17] R. Els and B. Monien, "Load Balancing of Unit Size Tokens and Expansion Properties of Graphs," *Proc. 15th Ann. ACM Symp. Parallel Algorithms and Architectures,* pp. 266-273, 2003.

[18] B. Ghosh, F.T. Leighton, B.M. Maggs, S. Muthukrishnan, C.G. Plaxton, R. Rajaraman, A. Richa, R.E. Tarjan, and D. Zuckerman, "Tight Analyses of Two Local Load Balancing Algorithms," *SIAM J. Computing,* vol. 29, no. 1, pp. 29-64, 1999.

[19] M. Litzkow, M. Livny, and M. Mutka, "Condor—A Hunter of Idle Workstations," *Proc. Eighth Int'l Conf. Distributed Computing Systems,* June 1988.

[20] R. Luling and B. Monien, "A Dynamic Distributed Load Balancing Algorithm with Provable Good Performance," *Proc. Fifth Ann. ACM Symp. Parallel Algorithms and Architectures,* pp. 164-172, 1993.

[21] O. Kremien and J. Kramer, "Methodical Analysis of Adaptive Load Sharing Algorithms," *IEEE Trans. Parallel and Distributed Systems,* vol. 3, no. 6, pp. 747-760, 1992.

[22] V. Robbert and B. Kenneth, "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining," citeseer.ist.psu.edu/vanrenesse01astrolabe.html, 2001.

[23] M. Balazinska, H. Balakrishnan, and D. Karger, "INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery," citeseer.ist.psu.edu/701773.html, 2002.

[24] D. Spence and T. Harris, "Xenosearch: Distributed Resource Discovery in the Xenoserver Open Platform," *Proc. 12th IEEE Int'l Symp. High Performance Distributed Computing (HPDC '03),* pp. 216-225, 2003.

[25] "Netmodeler Graph Library," http://boykin.acis.ufl.edu/wiki/index.php/Netmodeler, 2007.

[26] M. Mitzenmacher, "A Brief History of Generative Models for Power Law and Lognormal Distributions," *Internet Math. 1,* no. 2, pp. 226-251, 2003.

[27] Y. Azar, A.Z. Broder, A.R. Karlin, and E. Upfal, "Balanced Allocations," *SIAM J. Computing,* vol. 29, no. 1, pp. 180-200, citeseer.ist.psu.edu/article/azar94balanced.html, 2000.

Jesse S.A. Bridgewater received the PhD degree in electrical engineering from the University of California, Los Angeles (UCLA) in 2006. He is a postdoctoral research engineer with the Electrical Engineering Department at (UCLA). His research interests include distributed computing, peer-to-peer networking, and randomized algorithms.

P. Oscar Boykin received the PhD degree in physics from the University of California, Los Angeles. He is an assistant professor of electrical and computer engineering at the University of Florida. His research interests include fault tolerant computing, quantum cryptography, quantum information and computation and peer-to-peer networking. He is a member of the IEEE.

Vwani P. Roychowdhury received the PhD degree in electrical engineering from Stanford University. He is a professor of electrical engineering at the University of California, Los Angeles. His research focuses on computation models, including parallel and distributed processing systems, quantum computation and information processing, and circuits and computing paradigms for nanoelectronics and molecular electronics.