Virtualization And Self-organization For Utility Computing

Mehdi Saleh
University of Central Florida

VIRTUALIZATION AND SELF-ORGANIZATION FOR UTILITY COMPUTING

by

MEHDI SALEH

Bachelor of Science in Electrical Engineering

Sharif Institute of Technology, 2009

A thesis submitted in partial fulfillment of the requirements

for the degree of Master of Science Electrical Engineering

in the Department of Electrical Engineering and Computer Science

in the College of Engineering and Computer Science

at the University of Central Florida

Orlando, Florida

Spring Term

2011

Major Professor: Dan Marinescu

ABSTRACT

We present an alternative paradigm for utility computing when the delivery of service is subject to binding contracts; the solution we propose is based on resource virtualization and a self-management scheme. A virtual cloud aggregates set virtual machines to work in concert for the tasks specified by the service agreement. A first step for the establishment of a virtual cloud is to create a scale-free overlay network through a biased random walk; scale-free networks enjoy a set of remarkable properties such as: robustness against random failures, favorable scaling, and resilience to congestion, small diameter, and average path length. Constrains such as limits on the cost of per unit of service, total cost, or the requirement to use only “green" computing cycles are then considered when a node of this overlay network decides whether to join the virtual cloud or not.

A VIRTUAL CLOUD consists of a subset of the nodes assigned to the tasks specified by a Service Level Agreement, SLA, as well as a virtual interconnection network, or overlay network, for the virtual cloud. SLAs could serve as a congestion control mechanism for an organization providing utility computing; this mechanism allows the system to reject new contracts when there is the danger of overloading the system and failing to fulfill existing contractual obligations. The objective of this thesis is to show that biased random walks in power law networks are capable of responding to dynamic changes of the workload in utility computing.

This thesis is dedicated to my Mother,

who taught me that even the largest task can be accomplished if it is done one step at a time.

It is also dedicated to my Father,

who taught me that best kind of knowledge to have is that which is learned for its own sake.

ACKNOWLEDGMENT

Dr. Dan Marinescu has been the ideal thesis supervisor and instructor. His sage advice, insight criticisms, and patient encouragement aided the writing of this thesis in innumerable ways.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1: INTRODUCTION AND MOTIVATION

Utility computing is a generic term for packaging computation and storage as a service; the concept is inspired by public utilities which provide access to resources such as electricity, water, or communication bandwidth to the entire population. The illusion of infinite computing, the IT infrastructure are some of the attractions utility computing offers to individual users and to organizations that need low-cost access to computing resources. Utility computing is promoted vigorously by several companies aiming to exploit their expertise in information technology for providing low-cost, high-quality enterprise computing services; HP [23], Amazon [20], IBM, and Google are notable examples of companies invested in utility computing.

Cloud computing refers to an ensemble consisting of applications delivered as services, the so-called Software as a Service (SaaS), and the software and the hardware enabling data centers to offer these services. A survey of the state of the art of existing systems and of the classes of utility computing was conducted in the early 2009 by a group at U. C. Berkeley [4]. The authors of the study believe that the computation, communication, and storage models of the Amazon Web Services, Microsoft's Azure, and Google AppEngine ensure scalability and high availability of resources and discuss new opportunities in mobile interactive applications, parallel batch processing, decision support systems, and extension of compute-intensive desktop applications. While utility computing often requires a cloud-like infrastructure, its focus is on the business model on which providing the computing services are based. Cloud computing is a path to utility computing; other solutions may emerge in time, such as the one discussed in [35].

We distinguish on-the-spot requests for computing services, from services based on contracts; the first type is suitable for short-term, occasional requests from single users, while the second is demanded by large organizations which cannot afford to interrupt, or slow down their activity due to inadequate response to their computing needs. A contract, or a *Service Level Agreement*, SL, may specify: the elements necessary to determine the Class of Service (CoS), the minimum, average, and the maximum hourly/daily/weekly resource needs, the pattern of resource utilization, the response time, the range of service compliance indicators, and the penalties for failing to meet the contractual obligations. Contracts could be beneficial to users as well as providers of utility computing if an effective management system is in place; they can ensure QoS for the users and guide the long term investment policies of the providers of services. In this thesis we are only concerned with the second type of service demands and propose a new paradigm for the organization of utility computing.

To operate effectively, a provider of utility computing should minimize the long-term investments as well as the operating costs. The large peak-to-average resource requirements of individual applications may prevent the system from reaching optimal operation regions of the state space. The common answer to the unpredictability of the load of a system and of specific requirements is O*verprovisioning*; yet, this approach leads to long-term investments that cannot be justified. The alternative is to prevent congestion and reject new contracts which require immediate use of resources when the system operates near capacity; this approach emulates the congestion control mechanism in the Internet. To minimize the operating costs the organization providing utility computing could direct service requests to the sites with the lowest energy cost, redistribute the load and shut down systems when the load is light, or could redistribute the load to minimize the penalties when the system is overloaded and the QoS indicators could not be met.

The organization of a system providing utility computing is expected to respond to a set of often contradictory requirements. A centralized organization of a system with a very large number of components is problematic even if it is based on a hierarchical system; it is virtually impossible to accurately determine the global state of the system which as state and control information has to travel a long path between decision and execution sites. There is a general agreement that the management of a complex system should be automated, but the extent of the automation process is still debatable. While self-management is regarded as a highly desirable option, none of the existing systems for utility computing are based on self-management ideas; moreover, there are no comprehensive proposals or data supporting this approach.

Self-management is a facet of the broader concept of self-organization; though self-organization is difficult to define, its intuitive meaning is reflected in the observation made by Alan Turing that "global order can arise from local interactions." [48]. Inspired by biological systems, self-organization was proposed for networking [38] and even for economical systems [33]. Self-organization of biological systems is defined as "a process in which patterns at the global level of a system emerge solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed only with local information, without reference to global patterns". [12] Self-organization is used by different types of neural networks including Hopfield networks [27] and the networks proposed in [38]. The "swarm" algorithms [10], e.g., the Ant Colony Routing, mimic self-organization of social insects. Self-organization schemes have been proposed for *ad-hoc* and *sensor* networks [15], [37].

Virtualization is the process of simulating the interface to a physical object; traditionally, virtualization is based on multiplexing, on aggregation, or on emulation. In the first case, virtualization creates multiple virtual objects from one instance of a physical object; aggregation creates one virtual object from multiple physical objects; emulation constructs a virtual object from a different type of physical object. The separation of virtual from physical organization removes some of the characteristics and/or limitations of computing resources such as size, internal organization, reliability, or performance. For example, virtual memory enables the development of code independent of the size of the physical memory, Java Virtual Machine (JVM) permits the development of platform-independent code, threads allow sharing of a single processor, Redundant Array of Independent Disks (RAID) increase reliability, as well as, the performance of secondary storage devices. User Virtual Machines are an important element of the current architecture of computing clouds.

Virtualization, in the context of this paper, is based on multiplexing combined with aggregation. A processor is shared by multiple virtual machines; once an agreement between a provider and a customer is sealed a set of virtual machines cooperate to satisfy the conditions imposed by the agreement. In this paradigm virtualization supports self-management, enables the system to fulfill its contractual obligations, and, used judiciously, could contribute to lower costs. The solution we propose contrasts with the current organization of many cloud data centers where the virtual machines are clustered based on the computing needs of the consumers and each sub cluster is managed by one consumer.

We propose a probabilistic approach for resource virtualization based on biased random walks; the algorithm allows a subset of systems to create the overlay network interconnecting these systems. Monte Carlo methods are often used to solve optimization problems in multi-dimensional search spaces. One of the first applications of computers was based on an algorithm developed by Metropolis *et. Al* [39] for sampling in high dimensional probability distributions using Markov chains. This algorithm is at the heart of the strategy discussed in this paper; given a graph $G(V, E)$ let π be a strictly positive distribution on V and call k_i the degree of vertex I and $(i, j) \epsilon V$ the edge connecting vertices i and j. Then π is a stationary distribution of the Markov chain with transition probabilities

$$P_{i,j} = \begin{cases} \frac{1}{k_i} & if\ \frac{\pi_i}{k_i} \leq \frac{\pi_j}{k_j} \\ \frac{1}{k_j}\frac{\pi_j}{\pi_i} & if\ \frac{\pi_i}{k_i} > \frac{\pi_j}{k_j} \\ 1 - \sum_j p_{i,j} & if\ j = i \end{cases} \tag{1}$$

The work reported in this thesis is not restricted to a specific computing, communication, or storage model and it is complementary to the research on virtualization carried out now by several groups from industry [22], [23]. The virtualization architecture we propose is dynamic, virtual machines are created in response to an external request when local condition permit and have a limited lifetime. A virtual machine created on a node must act in concert with the other members of the virtual cloud and maintain only limited information about its neighbors in the overlay network. The virtual cloud created in response to SLA provides the services at a minimal cost and with minimal energy consumption.

The contributions of this thesis are: (i) Algorithms to dynamically create virtual clouds with a life-span determined by a contract between a user and the service provider. Individual systems join a virtual cloud based on local information regarding the available capacity, the cost to provide the service, and the energy consumption. (ii) A self-management scheme which takes advantage of desirable properties of the overlay network such as; small diameter, robustness to random failures, resilience to attacks, and scalability. In this scheme self-awareness can be archived at a small cost as individual systems are required to maintain information only about immediate neighbors.

Communication plays a critical role in any complex system and we start our analysis of virtual clouds with a discussion of the overlay network topology and we analyze biased random walks in chapter two. Analyzing the properties of scale-free-networks in chapter three. Virtual clouds and the distributed algorithms for their construction and the simulation studies in chapter four and five. In chapter six we discuss the results of dynamic workload distribution and we give a summary and discuss future work in chapter seven.

2: BIASED RANDOM WALKS & GRAPH ENTROPY

A strategy used successfully to locate systems satisfying a set of conditions in applications such as peer-to-peer systems is based on biased random walks; random walks are reported to be more efficient in searching for nodes with desirable properties than other methods such as flooding [21].

Unfortunately, the application of random walks in a large network with an irregular topology is unfeasible because a central authority could not maintain accurate information about a dynamic set of members. A solution is to exploit the fact that sampling with a given probability distribution can be simulated by a discrete-time Markov chain; indeed consider an irreducible Markov chain with states (i,j) $\in$ {0,1,…,S} and let $\mathcal{P}$ = *[Þij]* denote its probability transition matrix where

$$p_{ij} = Prob[X(t+1) = j \mid X(t) = i] \tag{2}$$

with X(t) the state at time *t*. Let $\pi = (\pi_0, \pi_1, \dots, \pi_S)$ be a probability distribution with nonzero probability for every state, $\pi_i > 0, 0 \leq i \leq S$. The transition matrix $\mathcal{P}$ is chosen so that π is its unique stationary distribution thus, the reversibility condition $\pi = \pi\mathcal{P}$ holds. When *g*(.) is a function defined on the states of the Markov channel and we wish to estimate

$$E = \sum_{i=0}^{S} g(i)\pi_i \tag{3}$$

We can simulate the Markov chain at times *t = 1,2,…, N* and the quantity

$$\hat{E} = \sum_{i=1}^{N} \frac{f(X(t))}{N} \tag{4}$$

is a good estimate of E, more precisely $\hat{E} \to E$ when $N \to \infty$. Hasting [26] generalizes the sampling method of Metropolis [39] to construct the transition matrix given the distribution π. He starts by imposing the reversibility condition

$$\pi_i \, p_{ij} = \pi_j \, p_{ji} \tag{5}$$

If $Q = [q_{ij}]$ is the transition matrix of an arbitrary Markov chain on the states {0,1,…,S} it is assumed that

$$p_{ij} = q_{ij} \alpha_{ij} \ if \ i \neq j \ and \ p_{ii} = 1 - \sum_{j \neq i} p_{ij} \tag{6}$$

Two version of sampling are discussed in [26], the one of Metropolis and one proposed by Baker [6]; the quantities α_{ij} are respectively:

$$\alpha_{ij}^{M} = \begin{cases} 1 & if \ \frac{\pi_j}{\pi_i} \geq 1 \\ \frac{\pi_j}{\pi_i} & if \ \frac{\pi_j}{\pi_i} < 1 \end{cases} \tag{7}$$

$$\alpha_{ij}^{B} = \frac{\pi_j}{\pi_i + \pi_j} \tag{8}$$

For example, consider a Poisson distribution $\pi_i = \lambda^i e^{(-\lambda)/i!}$; we choose $q_{ij} = \frac{1}{2}$ if j = i -1, i ≠ 0 or j = i + 1, i ≠ 0 and $q_{00} = q_{01} = 1/2$. Then using Baker's approach we have

$$p_{ij} = \begin{cases} \lambda / (\lambda + i + 1) & if \ \ j = i + 1, i \neq 0 \\ i / (i + \lambda) & if \ \ j = i - 1, i \neq 0 \end{cases} \tag{9}$$

and $p_{00} = 1/2$ and $p_{01} = \lambda e^{-\lambda} / (1 + \lambda e^{-\lambda})$. (10)

The algorithm to construct scale-free overlay topologies with an adjustable exponent in [46] adopts the equilibrium model discussed in [24]. The algorithm is based on random walks in a connected overlay network G(V,E) viewed as a Markov chain with state space V and a stationary distribution with a random walk bias configured according to a Metropolis-Hastings chain [26]. Recall that in this case we assign a weight $p_i = i^{-\alpha},\ 1 \leq i \leq N, \alpha \epsilon\ [0,1)$ to each vertex and add an edge between two vertices a and b with probability ${p_a}/{\sum_{i=1}^{N} p_i} \times {p_b}/{\sum_{i=1}^{N} p_i}$ if non exists they repeat the process until *mN* edges are created and the mean degree is 2*m*. Then the degree distribution is

$$p(k) \sim k^{-\gamma},\ \ with\ \ \gamma = \frac{1+\alpha}{\alpha} \tag{11}$$

The elements of the transition matrix $P = [p_{ij}]$ are

$$p_{ij} = \begin{cases} \frac{1}{k_i}\min\left\{\left(\frac{1}{j}\right)^{\frac{1}{\gamma-1}}\frac{k_i}{k_j}, 1\right\} & (i,j) \in E \\ 1 - \frac{1}{k_i}\sum_{(l,i)\in E} P_{il} & i = j \\ 0 & (i,j) \notin E \end{cases} \tag{12}$$

with k_i the degree of vertex i. An upper bound for the number of random walk steps can be determined from a lower bound for the second smallest eigenvalue of the transition matrix, a non-trivial problem.

We conclude that it is feasible to construct a scale-free global overlay network Ω network using a biased random walk algorithm. Though algorithms to detect phase transitions in a cloud are not discussed in this thesis we mention that the degree, γ, of the power law is related to phase transitions [45]; a small variation of γ can lead to an abrupt change in the macroscopic system behavior, e.g., its susceptibility to epidemics and resistance to failure. The *m*-th moment of the power law distribution of a discrete random variable X, $P_X(x = k) = k^{-\gamma}$ is

$$E[X^m] = \sum_{k=1}^{\infty} k^m P_X(x = k) = \sum_{k=1}^{\infty} k^m k^{-\gamma} = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-m}} \tag{13}$$

The first moment $E[X] = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-1}}$ diverges for $\gamma < 2$ and is identical to the Riemann's zeta function $\xi(\gamma - 1)$ for $\gamma \in (2, \infty)$; thus, in this range the average vertex degree is limited by a small constant. The variance $E[X^2] = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-2}}$ is divergent for $\gamma \leq 3$. The moments of a power law distribution play an important role in the behavior of a network. It has been shown that the giant connected component (GCC) of networks with a finite average vertex degree and divergent variance can only be destroyed if all vertices are removed; thus, such networks are highly resilient against faulty constituents [40].

Once the global overlay network Γ is constructed we wish to use biased random walks again to build Γ_C the overlay network for a virtual cloud. Now one of the core nodes of Γ initiates the construction of the power law interconnection network of the virtual cloud; the individual systems populating the vertices of the graph are subject to additional constraints regarding the distribution of the free capacity.

To analyze the implication of this strategy we discuss briefly *graph entropy.* Informally graph entropy is a measure of the degree of the randomness in a graph. Let G(V,E) be a directed graph with the set V of nodes of cardinality $n = |V|$ and $E \subseteq V \times V$ the set of edges. Assume that each node $j \in V$ in G has assigned a stochastic variable x_j selected from a state space (or finite alphabet) A with $s \geq 2$ elements. For every probability distribution p on the *n*-tuples $(x_1, x_2, \dots, x_n) \in A^n$ the entropy function $H_p(S)$ is given by:

$$H_p(S) = \sum_{v \in A^n} p(S,v)\, log_s \left(\frac{1}{p(S,v)}\right) \tag{14}$$

where *p(S,v)* for $v = (v_1, v_2, \dots, v_n) \in A^n$ is the probability that a tuple $(x_1, x_2, \dots, x_n) \in A^n$ is selected with $x_{s_1} = v_{s_1}, x_{s_2} = v_{s_2}, \dots, x_{s_n} = v_{s_n}$ where $S = \{s_1, s_2, \dots, s_n\}$.

If *H* denotes any entropy function *Hp* then *H* is normalized and *H(j) ≤ 1,* $\forall j \in \{1,2,\dots,n\}$ as the logarithms is in base *s*. It is easy to see that

$$H(j \mid i_1, i_2, \dots, i_n) = 0 \tag{15}$$

where $(i_1, j), (i_2, j), \dots, (i_n, j) \in E$ are edges with the head node *j*. indeed this equation states that there is no uncertainty of the value of the variable x_j if we are given the values of all the stochastic variables associated with the predecessor vertices of *j*.

The private entropy *E(G,s)* of a graph *G* over a state space A of size $s \in \{2,3,4,\dots\}$ is the supremum of Hp (1,2,…,n) of all entropy functions Hp over A that satisfy the n information constrains determined by G. The entropy E(G) is the supremum of E(G,s) for $s \in \{2,3,4,\dots\}$.

Clearly, the entropy of a graph with a power-law degree distribution is lower than that of a random graph, the graph has less randomness; this implies that a random walk in such a network is more constrained and the number of steps to select the desired number of nodes will be larger. Indeed our experiments showed that the number of steps to select 1000 nodes by a biased random walk is 40% larger in a graph with a power-law degree distribution than in a random graph.

The benefits of a well structured overlay network for a virtual cloud are more important than the time it takes to set it up; moreover, a virtual cloud typically includes a very small fraction of the number of systems in the cloud thus, our approach seems reasonable. On the other hand, time plays a critical role when we need additional resources for a virtual cloud.

3: POWER LAW DEGREE DISTRIBUTION AND SCALE-FREE NETWORKS

The topology of a network used to model the interactions in complex biological, social, economic and computing systems is described by means of graphs where vertices represent the entities and the edges represent their interactions. The number of edges incident upon a vertex is called the *degree of the vertex*.

Several models of graphs have been investigated starting with the Erdös-Reny model [19],[20] where the number of vertices is fixed and the edges connecting vertices are created randomly; this model produces a homogeneous network with an exponential tail, connectivity follows a Poisson distribution peaked at the average degree $\bar{k}$ and decaying exponentially for $k \gg \bar{k}$. An evolving network, where the number of vertices increases linearly and a newly introduced vertex is connected to *m* existing vertices according to a preferential attachment rule is described by Barabasi and Albert in [1], [2], [3], [7].

Regular graphs where a fraction of edges are rewired with a probability *p* have been proposed by Watts and Strogatz and called small-worlds networks [50]. Networks, whose degree distribution follows a power law, $P(k) \sim k^{-\gamma}$ are called <u>Scale-Free networks</u>. The four models are sometimes referred as ER (Erdös-Reny), BA (Barabasi – Albert), WS (Watts – Strogatz), and SF (Scale-Free) models, respectively [24]. BA networks with aging are investigated in [17]; a new site of the network is connected to some old site with probability proportional to the connectivity of the old site as in the BA model and to $\tau^{-\alpha}$ where *t* is the age of old site. The conclusion is that the network shows a scaling behavior only when $\alpha < 1$.

A number of studies have shown that scale-free networks have remarkable properties such as: robustness against random failures [8], favorable scaling [1], [2], [17], resilience to congestion [24], tolerance to attacks [47], small diameter [14] and average path length [7]. These properties make scale-free networks very attractive for interconnection networks in many applications including social systems [42], peer-to peer systems [45], sensor networks [36], [37] and, as we will argue in this thesis, to utility computing.

Consider and Erdös-Reny (ER) graph G_{ER} with *N* vertices; vertex *i* has a unique label from a compact set $i \in \{ 1 , \dots , N \}$. We wish to rewire this graph and produce a new graph G_{SF} where the degrees of the vertices follow a power-law distribution. The procedure we discuss consists of the following steps [34]:

1) We assign to each node *i* a probability

$$p_i = \frac{i^{-\alpha}}{\sum_{j=1}^{N} j^{-\alpha}} = \frac{i^{-\alpha}}{\zeta_N(\alpha)} \text{ with } 0 < \alpha < 1 \tag{16}$$

$$\text{And } \quad \zeta_N(\alpha) = \sum_{j=1}^{N} j^{-\alpha} \tag{17}$$

2) We select a pair of vertices *i* and *j* and create an edge between them with probability

$$p_{ij} = p_i p_j = \frac{(ij)^{-\alpha}}{{\zeta_N}^2(\alpha)} \tag{18}$$

And repeat these process n times.

Then the probability that a given pair of vertices *i* and *j* is not connected by an edge h_{ij} is

$${p_{ij}}^{NC} = (1 - p_{ij})^n \approx e^{-2np_{ij}} \tag{19}$$

And the probability that they are connected is

$${p_{ij}}^{C} = \left(1 - {p_{ij}}^{NC}\right) = 1 - e^{-2np_{ij}} \tag{20}$$

Call k_i the degree of vertex i; then the moment generating function of k_i is

$$g_i(t) = \prod_{j \neq i} [{p_{ij}}^{NC} + t{p_{ij}}^{C}] \tag{21}$$

The average degree of vertex i is

$$\bar{k}_i = t\,\frac{d}{dt} g_i(t)\big| t = 1 = \sum_{j \neq i} {p_{ij}}^{C} \tag{22}$$

Thus,

$$\bar{k}_i = \sum_{j \neq i} (1 - e^{-2np_{ij}}) = \sum_{j \neq i} \left(1 - e^{-2n\frac{(ij)^{-\alpha}}{{\zeta_N}^2(\alpha)}}\right)$$

$$\approx \sum_{j \neq i} 2n\frac{(ij)^{-\alpha}}{{\zeta_N}^2(\alpha)} = \frac{2n}{{\zeta_N}^2(\alpha)} \sum_{j \neq i} (ij)^{-\alpha} \tag{23}$$

This expression can be transformed as

$$\bar{k}_i = \frac{2n}{{\zeta_N}^2(\alpha)} \sum_{j \neq i} (ij)^{-\alpha} = \frac{2ni^{-\alpha} \sum_{j \neq i} j^{-\alpha}}{{\zeta_N}^2(\alpha)} = \frac{2ni^{-\alpha}(\zeta_N(\alpha) - i^{-\alpha})}{{\zeta_N}^2(\alpha)} \tag{24}$$

The moment generating function of k_i can be written as

$$g_i(t) = \prod_{j \neq i} [{p_{ij}}^{NC} + t{p_{ij}}^{C}] = \prod_{j \neq i} e^{-(1-t){p_{ij}}^{C}} = e^{(1-t)\sum_{j \neq i} {p_{ij}}^{C}} = e^{(1-t)\bar{k}_i} \tag{25}$$

Then we conclude that the probability that $k_i = k$ is given by

$$p_{d,i}(k) = \frac{1}{k!}\frac{d^k}{dt^k} g_i(t)| t = 0 \approx \frac{\bar{k}_i}{k!} e^{-\bar{k}_i} \tag{26}$$

When $N \to \infty$ then $\zeta_N(\alpha) = \sum_{i=1}^{N} i^{-\alpha}$ converges to the Riemann zeta function $\zeta_N(\alpha)$ for $\alpha > 1$ and diverges as $\frac{N^{1-\alpha}}{1-\alpha}$ if $0 < \alpha < 1$. For $0 < \alpha < 1$ equation (1) becomes

$$p_i = \frac{i^{-\alpha}}{\zeta_N(\alpha)} = \frac{1-\alpha}{N^{1-\alpha}} i^{-\alpha} \tag{27}$$

When $N \to \infty$, $0 < \alpha < 1$, and the average degree of the vertices is *2m*, then the degree of vertex i is

$$k = p_i \times mN = 2mN \frac{1-\alpha}{N^{1-\alpha}} i^{-\alpha} = 2m(1-\alpha)\left(\frac{i}{N}\right)^{-\alpha} \tag{28}$$

Indeed, the total number of edges in graph is *mN* and the graph has a power law distribution. Then

$$i = N\left(\frac{k}{2m(1-\alpha)}\right)^{-\frac{1}{\alpha}} \tag{29}$$

From this expression we see that there is a one-to-many correspondence between the unique label of the node *i* and the degree *k*; this reflects the fact that multiple vertices may have the same degree *k*. The number of vertices of degree *k* is

$$n(k) = N\left(\frac{k}{2m(1-\alpha)}\right)^{-\frac{1}{\alpha}} - N\left(\frac{k-1}{2m(1-\alpha)}\right)^{-\frac{1}{\alpha}} = N\left(\frac{k-1}{2m(1-\alpha)}\right)^{-\frac{1}{\alpha}}\left(\left(1+\frac{1}{k}\right)^{-\frac{1}{\alpha}} - 1\right) \tag{30}$$

We denote $\gamma = 1 + \frac{1}{\alpha}$ and observe that

$$\left(1+\frac{1}{k}\right)^{-\frac{1}{\alpha}} = 1 + \left(-\frac{1}{\alpha}\right)\left(\frac{1}{k}\right)^{-\frac{1}{\alpha}} + \frac{1}{2}\left(-\frac{1}{\alpha}\right)\left(-\frac{1}{\alpha}-1\right)\left(\frac{1}{k}\right)^{-\frac{1}{\alpha}-1} + \cdots \tag{31}$$

We see that

$$n(k) = N\left(\frac{(k-1)(\gamma-1)}{2m(\gamma-2)}\right)^{-\gamma+1} \times \left((1-\gamma)\left(\frac{1}{k}\right)^{-\gamma+1} - \frac{\gamma(1-\gamma)}{2}\left(\frac{1}{k}\right)^{-\gamma} + \cdots\right) \quad (32)$$

We conclude that the number of iterations to reach the value predicted by the theoretical model for the number of vertices of degree k is a function of *N*, of the average degree *2m*, and of γ, the degree of the power law. Next section discusses the properties of scale-free networks.

Scale-Free Networks:

The degree distribution of scale-free networks follows a power law; we only consider the discrete case when the probability density function $p(k) = af(k), f(k) = k^{-\gamma}$, and the constant a is $a = {}^{1}/_{\zeta}\,(\gamma, k_{min})$ thus

$$p(k) = \frac{1}{\zeta(\gamma, k_{min})} k^{-\gamma} \quad (33)$$

In this expression k_{min} is the smallest degree of any vertex, and for the applications we discuss in this thesis $k_{min} = 1;$ is the Hurwitz zeta function

$$\zeta(\gamma, k_{min}) = \sum_{n=0}^{\infty} \frac{1}{(k_{min} + n)^{\gamma}} = \sum_{n=0}^{\infty} \frac{1}{(1+n)^{\gamma}} \quad (34)$$

A scale-free network is *non-homogeneous*; the majority of the vertices have a low degree and only a few vertices are connected to a large number of edges. On the other hand, an exponential network is homogeneous as most of the vertices have the same degree. Another important property is that the majority of the vertices of a scale-free network are directly connected with the vertices with the highest degree; for example, in a network with N = 130 vertices and m = 215 edges 60% of the nodes are directly connected with the five vertices with the highest degree, while in an exponential network fewer than half, 27%, have this property [2]. The average distance *d* between the *N* vertices, also referred to as the diameter of the scale-free network scales as *ln N* in fact it has been shown that when $k_{min} > 2$ a lower bound on the diameter of a network with $2 < \gamma < 3$ is *N ln N* [14].

We now discuss a property of SF networks, the universal load distribution [24]. The load distribution in a directed or undirected SF network follows a power law with the exponent $\delta \approx 2.2$ insensitive to different values of γ in the range, $2 < \gamma < 3$. To reach this conclusion the authors of [24] assign a weight $p_i = i^{-\propto}, 1 \leq i \leq N, \propto \in [0,1)$ to each vertex and add an edge between two vertices a and b with probability ${p_a}/{\sum_{i=1}^{N} p_i} \times {p_b}/{\sum_{i=1}^{N} p_i}$ if none exists; they repeat the process until *mN* edges are created and the mean degree is 2*m*. Then the degree distribution is

$$p(k) \sim k^{-\gamma},\ with\quad \gamma = \frac{(1+\alpha)}{\alpha}\ \ and\ \ \alpha = \alpha\gamma - 1. \tag{35}$$

The probability p_i for vertex *i* can be expressed as

$$p_i = i^{-\alpha} = i^{\frac{1}{\alpha\gamma - 1}} \tag{36}$$

When the load l_i at vertex i is defined as the total amount of data packets passing through the vertex when all pairs of vertices send and receive one data packet between them, then numerical simulations show that the load distribution follows also a power law when $2 < \gamma < 3$

$$p_{load}(k) \sim k^{-\delta} \text{ with } \beta \approx 0.8 \text{ and } \delta \approx 1 + {1}/{\beta} = 2.2. \tag{37}$$

The condition $2 < \gamma < 3$ implies that $1/2 < \alpha < 1$. The communication load is proportional with the degree of a vertex

$$l_i \sim k^{\frac{\gamma-1}{\delta-1}} \tag{38}$$

[1]The Hurwitz zeta function $\zeta(s,q) = \sum_{n=0}^{\infty} \frac{1}{(q+n)^s}$ for $s,q \in \mathbb{C}$ and $Re(s) > 1$ and $Re(q) > 0$. The Riemann zeta function is ζ(s,1).

Thus, the load at each vertex is directly proportional with its degree if and only if $\gamma = \delta$

The communication load at vertex i with degree k and the total communication load are, respectively,

$$l_i = c(NlogN)(N/k)^{\beta} \quad and \quad L = \sum_{i=1}^{N} l_i = c' N^2 logN \text{ with } c \text{ and } c' \text{constants.} \tag{39}$$

The degree, γ , of the power law is related to phase transitions [45]; a small variation of it can lead to an abrupt change in the macroscopic system behavior, e.g., susceptibility to epidemics and resistance to failure. The *m-th* moment of the power law distribution of a discrete random variable *X*,

$$E[X^m] = \sum_{k=1}^{\infty} k^m P_X(x = k) = \sum_{k=1}^{\infty} k^m k^{-\gamma} = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-m}} \tag{40}$$

The first moment $E[X] = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-1}}$ diverges γ < 2 and is identical to the Riemann's Zeta function (γ – 1) for $\gamma \in (2, \infty)$, thus, in this range the average vertex degree is limited by a small constant. The variance $E[X^2] = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-1}}$ is divergent for $\gamma \leq 3$. The moments of a power law distribution play an important role in the behavior of a network. It has been shown that the giant connected component (GCC) of networks with a finite average vertex degree and divergent variance can only be destroyed if all vertices are removed, thus, such networks are highly resilient against faulty constituents [40].

Epidemic or gossip algorithms are often used in communication to accomplish tasks such as: (i) disseminate information, e.g., topology information; (ii) compute aggregates, e.g., arrange the nodes in a gossip overlay into a list sorted by some attributes in logarithmic time; or (iii) manage data replication in a distributed system [25], [29], [30]. The epidemic threshold λ for the *Susceptible-Infectious–Recovered* (SIR) [31] and *Susceptible–Infectious–Susceptible* (SIS) [11] epidemic models of power networks can be expressed as $\lambda = \frac{E[X]}{E[X^2]}$ the epidemic threshold is defined as the minimum ratio of infected nodes to the cured nodes per time such that it still allows the epidemics to continue without outside infections. It follows that $\lambda \mapsto 0$ if $\gamma \in (2,3)$; in other words; such networks become infinitely susceptible to epidemic algorithms. This property is very important for dissemination of control information; also self-awareness requires constant monitoring of other vertices possibly using epidemic algorithms.

Though our discussion is focused on scale-free overlay networks to support service-level agreements, it seems reasonable to consider that the physical cloud itself could be logically interconnected through a scale-free overlay network. Indeed, the cloud could consist of a very large number of physical units $N \geq 10^8$, located at several sites and interconnected by high-speed networks. The relatively few high-degree nodes of the overlay network organized as a power law distribution could monitor subsets of nodes at a small distance from each and could gather performance data for determining the impact of management policies e.g., utilization of resources (CPU, communication bandwidth, storage, etc.), power consumption, synthetic indicators regarding the quality of service (missed deadlines, inability to ensure the capacity required by SLAs, etc.). The global organization of the physical cloud as a scale free network is important for detecting undesirable global phenomena such as phase transitions; early detection of phase transitions could result in preventive measures or, in the worst case, to a controlled transition to a different operating regime when only the most important activities are carried to completion.

Vulnerability to attacks could be a problem for networks where the degree of the nodes follows a power law distribution; indeed, the diameter of these networks increases rapidly when the most connected nodes are targeted; under an attack a large network could break into isolated fragments. But again we can exploit the fact that a very small fraction of the nodes have a high degree of connectivity. We can estimate the number of "core nodes," nodes with a degree larger than k_{lim} when the scale-free network consist of N nodes

$$N_{lim} = N \times \text{Prob}\,(k \geq k_{lim}) \qquad (41)$$

$$\text{with} \quad \text{Prob}(k \geq k_{lim}) = 1 - \text{Prob}(k < k_{lim}) = 1 - \sum_{k=1}^{k_{lim}} p(k) \qquad (42)$$

We can replicate the P_{lim} critical nodes, each replica mirroring the role of the primary node thus, not being a member of the logical organization of the overlay network; as $\text{Prob}\,(k \geq k_{lim})$ is small we expect the additional cost to be justified by the increased resilience to attacks.

As an example we consider the case $\gamma = 2.5$ and $X_{min} = 1$, Table 1; we first determine the value of the zeta function $\zeta(\gamma, x_{min})$ and approximate $\zeta(2.5, 1) = 1.341$ thus, the distribution function is

$$p(k) = \frac{k^{-2.5}}{1.341} = 0.745 \frac{1}{k^{2.5}} \tag{43}$$

where k is the degree of each vertex.

The probability of vertices with degree $k > 10$ is $\text{Prob}(k > 10) = 1 - \text{Prob}(k \leq 10) = 0.015$. This means that at most 1.5% of the total number of vertices will have more than k edges connected to them; we also see that that 92.5% of the vertices have degree 1, 2 or 3.

TABLE I:

Table 1: A power law distribution with degree γ = 2.5; the probability, p(k), and n_k, the number of vertices with degree k, when the total number of vertices is N = 10^8.

K	*p(k)*	n_k
1	0.74	74.5×10^6
2	0.13	13.1×10^6
3	0.04	4.9×10^6
4	0.02	2.3×10^6
5	0.01	1.3×10^6
6	0.00	0.9×10^6
7	0.00	0.6×10^6
8	0.00	0.4×10^6
9	0.00	0.3×10^6
10	0.00	0.2×10^6

The question we address next is how to estimate the degree distribution of any scheme for the construction of a power-law network. The estimation of the degree distribution from empirical data is analyzed in [13]**;** according to this study a good approximation for a discrete power law distribution for a network with P vertices and k_{min} = 1 is

$$\hat{\gamma} \approx 1 + P\left[\sum_{i=1}^{P} ln \frac{k_i}{k_i - {}^1/_2}\right]^{-1} = 1 + \frac{P}{\sum_{i=1}^{P} 2k_i} \tag{44}$$

Several measures exist for the similarity/dissimilarity of two probability density functions of discrete random variables including the trace distance, fidelity, mutual information, and relative entropy [16], [32]. The *trace distance* (also called Kolmogorov or L1 distance) of two probability density functions, $p_X(x)$and $p_Y(y)$, and their *fidelity* are defined as

$$D\big(p_X(x), p_Y(x)\big) = \frac{1}{2}\sum_x \left|p_X(x) - p_Y(x)\right| \tag{45}$$

and $$F\big(p_X(x), p_Y(x)\big) = \sum_x \sqrt{p_X(x)p_Y(x)} \tag{46}$$

The trace distance is a metric: it is easy to prove non-negativity, symmetry, the identity of indiscernible, and the triangle inequality. On the other hand, the fidelity is not a metric, as it fails to satisfy the identity of indiscernible.

$$F\big(p_X(x), p_X(x)\big) = \sum_x \sqrt{p_X(x)p_X(x)} = 1 \neq 0 \tag{47}$$

Determining either the $L1$ distance between the distribution calculated is based on equation (18) and the one produced by the algorithm discussed in Section III requires information about the degree of all vertices.

From Table I we see that the degree-one vertices represent a very large fraction of the vertices of a power-law network. We wish to determine whether the number of degree-one vertices provides an adequate stopping criterion instead of the *L1*.

A recent paper [46] proposes a distributed rewiring scheme to construct scale-free overlay topologies with an adjustable exponent. An alternative method of creating of the scale-free overlay network could be based on the gossip-based peer-sampling discussed in [30].

4: VIRTUAL CLOUDS

We assume that the computing systems of the organization supporting utility computing are distributed across multiple sites and interconnected by high-speed and low-latency networks reliable networks. This assumption allows us to concentrate on overlay networks and exploit the properties of the logical organization of communication. A *virtual cloud* is a subset of the systems assigned to the tasks specified by a service-level agreement, as well as the overlay network interconnecting these systems. In a virtual cloud there is no central authority responsible for resource management; an individual node decides to join a virtual cloud based solely on local state information provided by the local workload manager [23] and by the local power management system.

A scale-free global overlay network supports communication among the large number of systems, $\sim 10^8$, of the organization providing utility computing. The lifespan of the global overlay network Γ is dictated by administrative considerations and it is expected to be of the order of days if not weeks. The degrees of the nodes of the global overlay network follow a power law distribution thus, the network is heterogeneous.

The core nodes of Γ are the ones with a degree $k \geq k_{limit}$; k_{limit} is a function of the number of systems in the organization providing utility computing and its user population. The n_c core nodes represent a very small fraction, $\frac{n_c}{n}$, of the total number of nodes, n; if n_1 denotes the number of nodes at distance one from a core node then the ratio $\frac{n_1}{n-n_c}$ is very close to unity as shown in chapter three.

The heterogeneity of this scale-free overlay network is exploited for self-management; the core node of Γ assume management functions, monitor the system, coordinate activities, and prevent phase transitions. They periodically collect information regarding the free capacity of nodes close to them; then, the core nodes exchange such information among them and construct an approximate distribution of the free capacity at time t_i with mean $\mu(t_i)$ and variance $\sigma(t_i)$.

Assuming that the information propagates from one node to another in one unit of time and that the core nodes form a ring topology to exchange control information then, at time t_i the n_c core nodes will have accurate information about the states of the n_1 systems at time t_{i-1}; it will take at most n_c units of time until all core nodes can construct an approximate distribution of the free capacity. Thus, the distribution of the free capacity available to all core nodes at time t_i reflects the state of the entire system at time $t_{i-(n_c+1)}$. The actual distribution at time t_i is slightly different that the one provided by the core nodes; we assume that its mean value $\acute{\mu}(t_i)$ is uniformly distributed around $\mu(t_i)$ and the variance $\acute{\sigma}(t_i)$ is also uniformly distributed around $\sigma(t_i)$.

Once the global overlay network Γ is constructed we use biased random walks to build Γ_c, the overlay network for a virtual cloud c. Now one of the core nodes of Γ which will be called the *SLA-Coordinator* (SLAC) initiates the construction of the overlay network of the virtual cloud c.

To determine the size of the overlay network ΓC for the virtual cloud c the SLAC examines the average and the maximum workload, chooses a distribution of the load and the number virtual cloud nodes and amplifies this number based on local information regarding the distribution of the free capacity, of the power consumption per unit of work, of the “green" nodes, and so on. Given the wide range of service demands the SLAC should have access to Ontology to track various aspects of performance monitoring of the SLA [19].

The overlay network Γ_c consists of a number of nodes larger than the number to be included in the virtual cloud c. The selection is a result of successive biased random walks in Γ; an individual node decides to join a virtual cloud c based solely on local state information provided by the local workload manager and by the local power management system. The criteria to join virtual cloud c could be: the free capacity of the node larger than a given threshold; the cost per unit of service below a certain limit; the node provides “green" computing cycles and so on. Successive random walks select increasingly smaller subsets of nodes that satisfy the additional constrains. As a result of this strategy the nodes included in the virtual clouds are a subset of the nodes of the overlay network; when additional resources are needed we expand the virtual cloud with nodes of the overlay network we have previously rejected.

The overlay network supports functions related to explicit application requirements specified by SLA and for the management of the activities, including workload distribution, system monitoring, error recovery, minimization of costs and reduction of power consumption, and so on. The self-awareness and self-repair properties of the virtual cloud benefit from the fact that the majority of nodes have degree one or two and have to maintain information about the role of one of two neighbors; when a node fails in most cases there is either one, or at most, two neighbors which attempt to trigger rewiring and this simplifies the rewiring strategy.

The basic architecture of the systems we propose is illustrated in Figure 1. Once a request to join a virtual cloud is received the MVM compares the available capacity and the future resource commitments with the load and the future resource commitments specified by the request and decides whether the system should join the new virtual cloud or not. When a decision to join a virtual cloud is taken then the Management Virtual Machines creates a Virtual Cloud Manager; the VCM will maintain the information regarding the topology of the virtual cloud and manage communication with the other members of the virtual cloud. Then the VCM creates the Guest Virtual Machine which will provide the user environment for the application specified by the SLA.

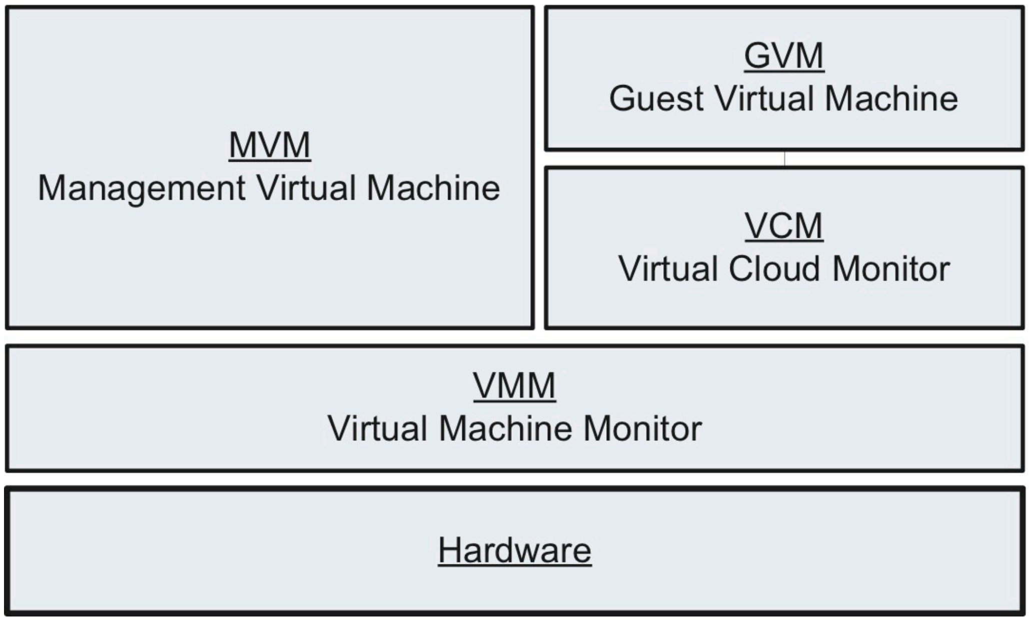


Figure 1: The virtualization architecture

5: DISTRIBUTED ALGORITHMS FOR THE FORMATION OF VIRTUAL CLOUDS

The algorithm to construct a power low network using a random walk require each one of the *N* nodes of the cloud to have a unique cloudId (cId) selected from a compact set of integers $1 \leq cId \leq N$. The *cId* could be assigned to each system by a central authority at the startup time; alternatively, each system could use the MAC address of one of its network interfaces and then we can run a distributed sorting of these addresses to obtain the unique *cId* in the range (1,*N*).

The core nodes of Ω are the ones with a degree $k \geq k_{limit}$; k_{limit} is a function of the size of the cloud and its user population. There are only a few high degree nodes thus; there is a *natural selection criteria* for core nodes. The core nodes assume management functions, monitor the system, coordinate activities, and prevent phase transitions; these nodes can be replicated to increase the resilience to attacks and make the system fault-tolerant. A large fraction of the nodes of the cloud are directly connected to the core nodes as discussed in Chapter three.

The core nodes should be able to exchange efficiently their local views regarding the state of the system. A self organization procedure to link the core nodes into a ring structure is sketched next. To be included in the ring a core node should discover its left and right core neighbors; initially a core node will set the *cId* of the left and the right partners as its own. Each core node is aware of the degree of each one of its neighbors and should send to the neighbors with a degree $k \geq 2$ an identification message with a hop count slightly larger than the diameter of the network and request that the message be forwarded to their neighbors of degree $k \geq 2$ until the hop count reaches zero. A core node will eventually receive identification messages from all other core nodes; upon receiving the message it will compare the *cId* of the message with the current *cId* of left and of the right partners; if it is smaller than the *cId* of the left partner, or larger than the one of the right partner, no action will be taken; if it is larger than the one of the left partner, but smaller than its own, it will replace the one of the left partner; if it is smaller than the one of the right partner, but larger than its own, it will replace the one of the right partner. Finally, the two nodes, one with no left partner and one with no partners will join; this can be done if each node maintains the larger and the smallest *cId* it has ever seen in an identification message.

The core nodes should estimate several probability distribution functions such as: the cost per unit of service $F_{Cost}(c)$, the green cycle $F_{Green}(g)$. Each core node would gather the information from the nodes at distance one and then pass this information to its right partner in the ring and eventually receive from its right partner a version incorporating a global view. This information is relatively stable and does not need to be updated frequently. A more complex process to estimate the distribution of the free capacity of the nodes must be designed; a possible solution is to require a core node to gather this information periodically from the virtual clouds it has created and then to exchange it with its core partners.

The second type overlay networks are the ones created dynamically for virtual clouds; typically, they have a relatively small number of nodes, $N^q \sim 10^2 - 10^4$, and a limited lifespan. The heterogeneity of a power law overlay network could be exploited for interconnecting physical systems located at sites in different geographic areas thus, minimizing the potential effect of a catastrophic event such as a blackout affecting a large geographic region. The properties of a scale-free overlay network can be exploited for self-management of a service-level agreement in a virtual cloud. The self-awareness and self-repair properties of the virtual cloud benefit from the fact that the majority of nodes have degree one or two and have to maintain information about the role of one of two neighbors; when a node fails in most cases there is either one, or at most, two neighbors which attempt to trigger rewiring and this simplifies the rewiring strategy.

We assume that we have constructed a scale-free global overlay network, Ω with $2 < \gamma < 3$, e.g., using the algorithm described in this section or the one in [46] and we discuss next how to construct a virtual cloud subject to several constraints in addition to an optimal overlay network; for example, an SLA could specify the cost the user is willing to pay for the services, request sites with a cost per unit of service lower than a given threshold, and require "green" computing cycles. We should include in the virtual cloud only systems whose free capacity at the time when the virtual cloud is created follows a certain distribution. For example, we may want to identify systems whose free capacity is uniformly distributed and then partition the workload for the SLA to guarantee load balancing for the systems included in the virtual cloud. An application may be naturally decomposed into tasks with a particular distribution of the workload for example, normal or Poisson distributions; in this case, the additional condition would be to include systems whose available capacity follows the distribution of the application's workload. Incidentally, construction of the virtual cloud could also support co-scheduling [5], in other words, guarantee that the systems included in the cloud start working on the tasks required by the SLA at the same time; co-scheduling is important for distributed applications that require barrier synchronization.

Given an SLA we construct first an overlay network consisting of a number of nodes larger than the number we wish to include in the virtual cloud; this selection process is done by a random walk in Ω when we select a node based on criteria such as: whether it provides "green" computing cycles or not; if its free capacity is larger than a given threshold; or if its cost per unit of service is below a certain limit. Then we select increasingly smaller subsets of nodes that satisfy the additional constrains through several random walks. As a result of this strategy the nodes included in the virtual clouds are a subset of the nodes of the overlay network; when additional resources are needed we expand the virtual cloud with nodes of the overlay network we have previously rejected. Recall that the nodes included in the power law network satisfy minimum requirements regarding the cost per unit of service and energy consumption, thus the expansion of the virtual cloud may not grantee optimality, but does not sacrifice the objectives to reduce cost and power consumption.

The process to create a virtual cloud is initiated by *A*, one of the core nodes of Ω which first parses SLA^q and determines critical parameters for the cloud including: the number of nodes N^q; the cost C^q; the parameters of the distribution of the free capacity of the nodes to be included in the cloud; the maximum power consumption per unit of service, g^q; and possibly other information. The buildup of a virtual cloud consists of the following steps:

Step 1. Select $2 < \gamma < 3$ and compute $\alpha = \frac{1}{\gamma - 1}$; a core node *A* initiates the creation of Ω^q, a scale-free network with

$$p(k) = \frac{1}{\zeta(\gamma,1)} k^{-\gamma} \tag{48}$$

and p_i, the probability of the vertex with $cId = i$ given by

$$p_i = i^{\frac{1}{\alpha\gamma - 1}} \tag{49}$$

Call Γ_1^q the set of nodes of Ω included in Ω^q and let N_1^q be the cardinality of Γ_1^q, $N_1^q = |\Gamma_1^q|$. If $\bar{k}$ denoted the average degree of a node of a power law network with the exponent γ then the number of edges$|E|$ of this overlay network is

$$|E| = \bar{k}\,\frac{N_1^q}{2} \text{ with } \bar{k} = \sum_{k=1}^{\infty} \frac{1}{k^{\gamma-1}} = \zeta(\gamma-1,0). \qquad (50)$$

Step 2. A random walk in Ω^q to select a subset of the nodes of Ω^q following a the distribution of the free capacity; call this subset $\Gamma_2^q \subset \Gamma_1^q$ and let $N_2^q < N_1^q$ be the cardinality of Γ_2^q.

Step 3. A random walk to select a subset $\Gamma_3^q \subset \Gamma_2^q$ to ensure that the total cost does not exceed C^q. The cardinality of Γ_3^q, satisfies the condition $N_3^q < N_2^q < N_1^q$. First we determine the average cost per node $\overline{c^q} = {c^q}/{N_4^q}$ and then select the nodes subject to the condition $c_i \le \overline{c^q}$

with c_i the cost per unit of service for node *i*.

Step 4. A random walk to select a subset $\Gamma_4^q \subset \Gamma_3^q$ of nodes that can provide the service with a power consumption per unit of service at or below the threshold e^q. The cardinality of Γ_4^q satisfies the condition $N_4^q < N_3^q < N_2^q < N_1^q$

It follows the additional constraints filter out systems; the number of nodes in successive subsets is given by

$$N_4^q = \Pr(g \le g^q) \times N_3^q = F_{greeen}(g^q) \times N_3^q \qquad (51)$$

$$N_3^q = \Pr(g \le g^q) \times N_2^q = F_{cost}(g^q) \times N_2^q \qquad (52)$$

$$N_2^q = \Pr(w > w^q) \times N_1^q = (1 - F_{free}(w^q) \times N_1^q \qquad (53)$$

To include in the virtual cloud N^q systems we determine N_1^q, the number of nodes of the overlay network from the equation

$$N^q = N_4^q = F_{green}(g^q) \times F_{cost}(g^q) \times \left(1 - F_{free}(w^q)\right) \times N_1^q \qquad (54)$$

Then we compute the number of edges

$$|E| = \zeta(\gamma - 1, 0) \frac{N_1^q}{2} \qquad (55)$$

The algorithm to generate the scale-free network Ω^q with N^q nodes and |E| edges consists of the following steps:

1) Use a random walk to select the subset Γ_1^q of nodes of Ω to be included in Ω^q. For example, select only "green" computing nodes, or nodes based upon their geographic location.

2) Assign each node a *vId*, $1 \leq vId \leq N^q$.

3) Set L the random walk length, e.g., $L = 10$.

4) Set the number of nodes already rewired, $n_{rewired} = 0$.

5) Select at random a node from Γ_1^q, e.g., node **a** and check if it has any edge that has not been rewired yet.

a) If NO go to step 5.

b) If YES pick up one of the edges at random and save both endpoints of that edge.

6) Check which one of the endpoints has higher degree, if they were same pick one of at random.

7) Initialize the number of hops for the random walk $n_{hop} = 0$.

8) Draw a random number $0 < r < 1$.

9) Pick up one at random a node in the neighborhood of the original node **a**, e.g. node **b** .

10) Given the degree d_a of node **a** with vId_a and the degree d_b of node **b** with vId_b, we calculate

$$h = \frac{d_a}{d_b}\left[\frac{vId_a}{vId_b}\right]^{\frac{1}{\propto\gamma-1}} \tag{56}$$

a) If $h > r$ send a message to node **b**.

b) If $h \leq r$ send a message to node **a**.

11) Increment the number of hops $n_{hop} = n_{hop} + 1$.

a) If $n_{hop} \neq L$ and $n_{hop} < 2L$ go to Step 8.

b) If $n_{hop} = L$ save the node as the target node **c** then go to Step 8.

c) Else save the node as the second target node **d**.

12) Connect target nodes to each other.

13) Remove the edge found in Step 5*b*.

14) Mark the edge you found as a rewired edge.

15) Increment the number of nodes already rewired,

$$n_{rewired} = n_{rewired} + 1 \tag{57}$$

a) If $n_{rewired} \leq E$ go to Step 5.

b) Else, the algorithm terminates as we have rewired all edges.

Once we have constructed the scale-free network Ω^q with N^q nodes and $|E|$ edges we proceed to select subsets of nodes based on additional restriction. For example, if the free capacity has a normal distribution we select a subset of N_2^q nodes according to the following algorithm:

1) Initialize the number of nodes processed $n^p = 0$

2) Pick up a random node and call it node **a**.

3) Pick up one of its neighbors at random and call it node **b**.

4) Draw a random number ($0 \leq r \leq 1$).

5) If C_a is the free capacity of node a and C_b is the free capacity of random neighbor we calculate

$$\beta = e^{-({C_a}^2 - {C_b}^2)} \tag{58}$$

a) If $\beta \ \geq r$ or $\beta \ \geq 1$ go to the neighbor.

b) then set **b** as the new node.

6) Else stay in the the same node **a**.

7) Increment the number of nodes, $n_p = n_p + 1$.

a) If $n_p < N_2^q$ go to step 2

b) Else terminate.

Uniform distribution on power remaining:

1) From 100 nodes that we've built the normal distribution, pick up a random node and call it **a**.

2) Find the most nearest power remaining in the nodes next to the random node, e.g. **b**.

3) If the $|P_a - P_b|$ is less than Th = 0.05 go to the node **b**.

4) Remove the original node e.g. **a**.

5) Set up the **b** as new node e.g. **a**.

6) Increment the number of hops, $n_p = n_p + 1$

a) If $n_p \ \leq 10$ go to step 2.

b) Else terminate.

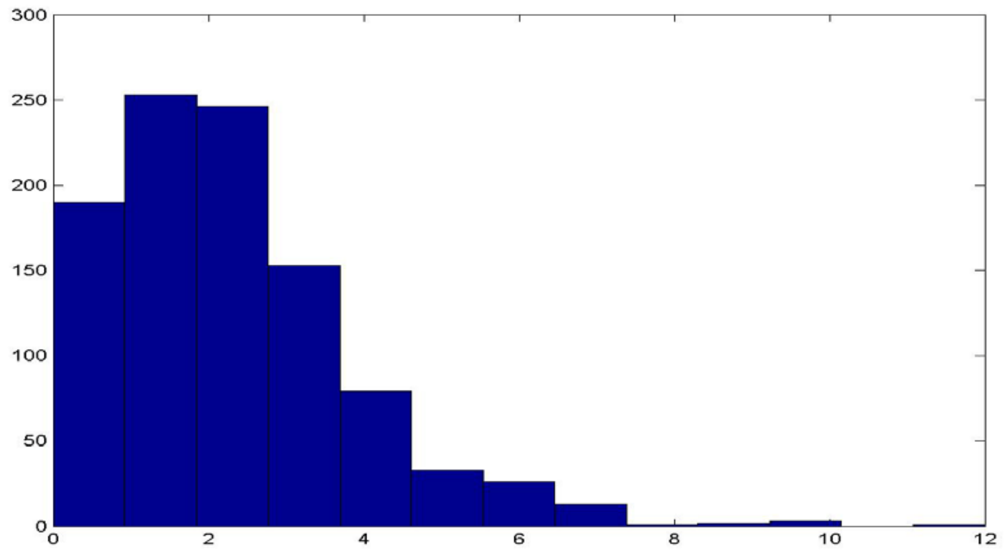
7) Else increase the Threshold 0.01 then go to step 2.

Simulation Studies:

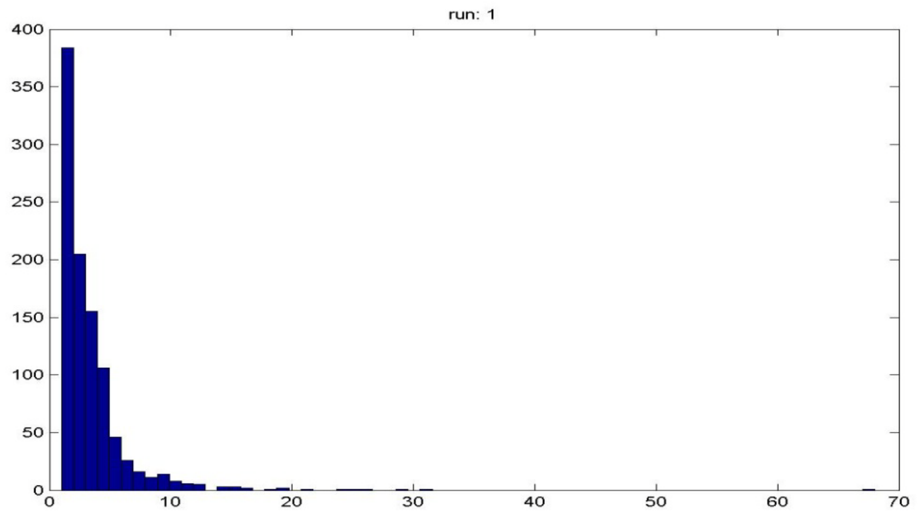
Development of a complex system is often based on intensive preliminary simulation studies; indeed, analytical performance studies of systems with a very large number of components is rarely feasible and the development of a test bed system can be prohibitively expensive. In this section we report on our simulation studies of the algorithms and strategies for the creation of virtual clouds; the simulation environments we used in our research on self-organization of sensor networks [36], [37] could not handle the requirements of this project and we developed a *C*-based simulator running on Linux workstations, as well as, Powerbooks under MAC OS.

A first question we address is if the procedure discussed in Chapter Three can be used in practice to produce a power law distribution; our experiments show that stopping after mN iterations leads to a distribution of the degrees of vertices resembling a power law, but not to the one we expected for $\gamma = 2.5$.

Figure 2(a) shows the histogram of the degree distribution for the random graph used as input to the algorithm; Figure 2(b) shows the histogram of the degree distribution after a number of iterations equal to the expected number of edges $mN = 1.5 \times 1000$ of the graph of the power law network. When $\gamma = 2.5$ we expect 745 vertices of degree-one for a network with $N = 1{,}000$ vertices; the actual number of degree-one vertices is ~ 380, far from the expected value. Recall from Chapter Four that degree-one vertices represent a very large fraction of the vertices of a power-law network this number can be used as an indicator of the similarity between the degree distribution obtained experimentally and the one expected for a power law distribution with the given the degree of the power law distribution, γ.

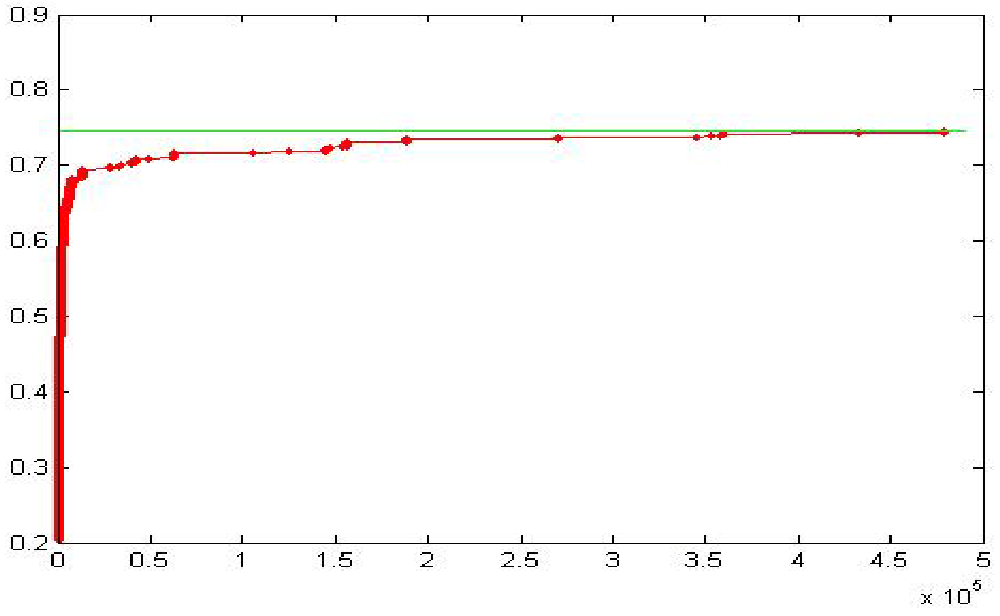


(a)

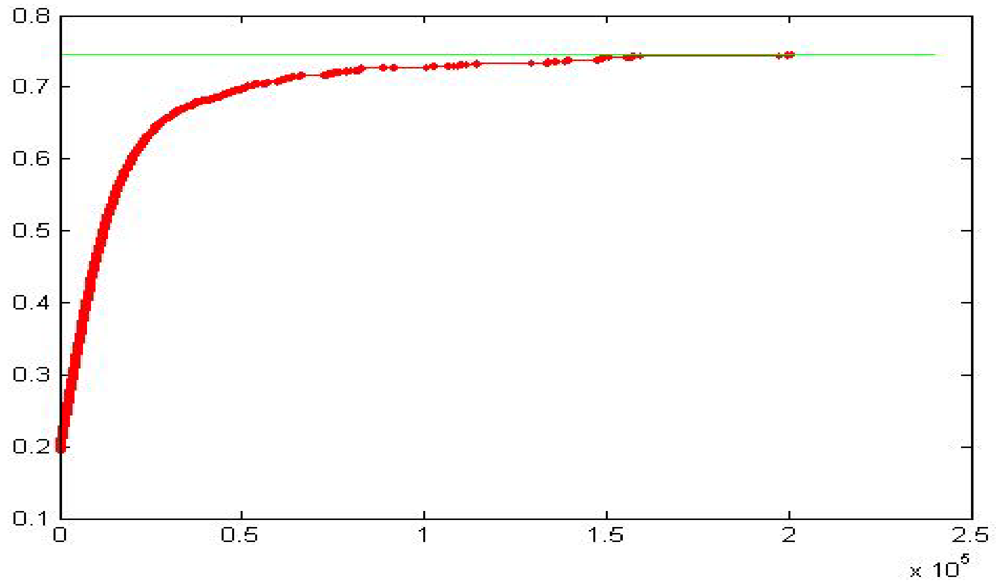


(b)

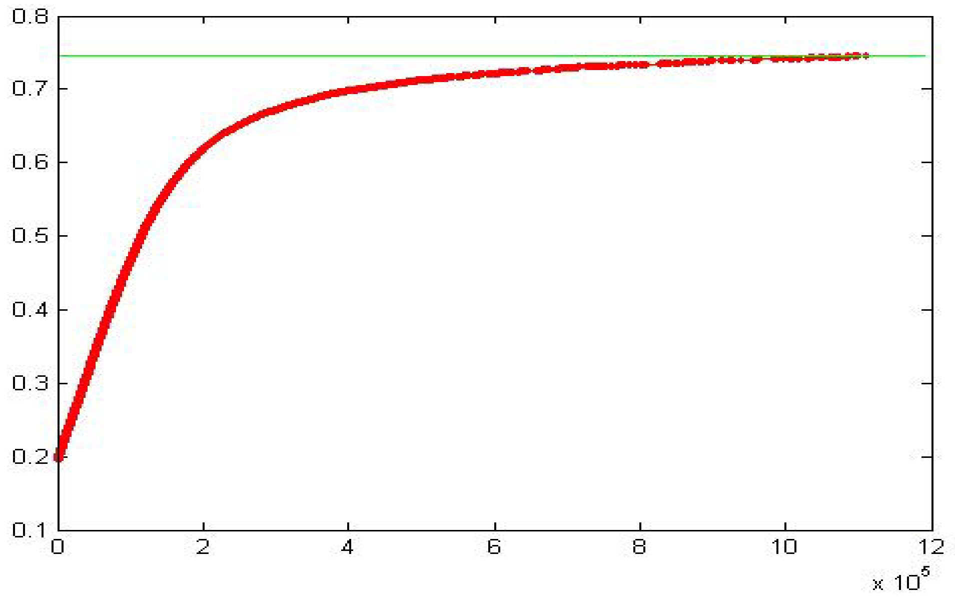
Figure 2: Two histograms of the degree distribution of: (a) the random graph with N = 1,000 vertices used as input to the algorithm in Chapter Three; (b) the graph after one run of the algorithm with mN = 1,500 iterations.



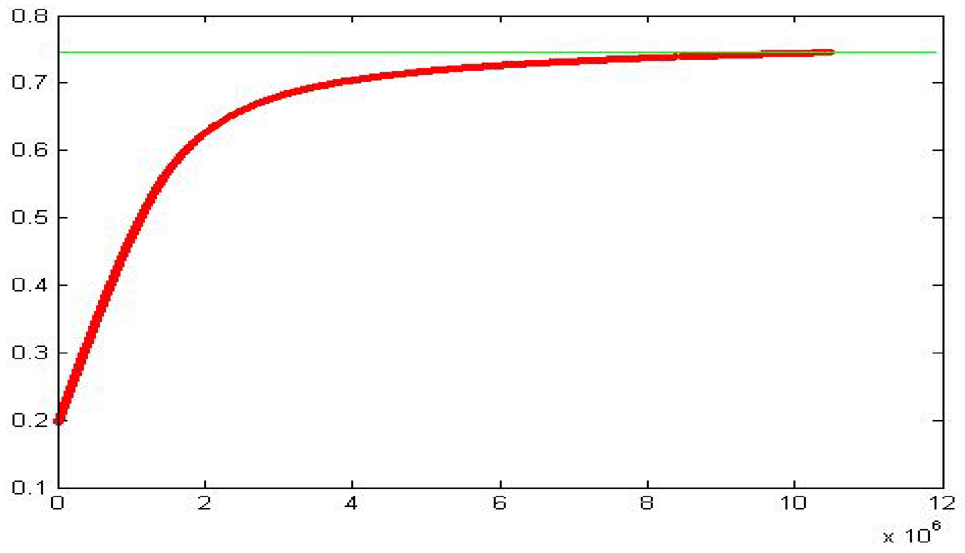
(a)



(b)



(c)



(d)

Figure 3: The fraction of degree-one vertices $\upsilon = N_1/N$ is a function of the number of iterations when $\gamma = 2.5$; the value calculated in Chapter Four is 74.5%. The number of iterations required for υ to be within 2% of the theoretical value is (a) 10×10^5 when N = 1,000 vertices; (b) 1.5×10^5 when

N = 1,0000 vertices; (c) 10 × 10^5 when N = 100,000 vertices; (c) 10 × 10^6 when N = 1,000,000 vertices.

Table 2 : The time required by the algorithm to construct the scale-free network to converge to the theoretical value for degree-one vertices is a function of N, the number of vertices.

Number Vertices	**Mean Execution Time (Sec)**	**Standard Deviation (Sec)**	**95% Confidence Interval for the Mean (Sec)**
1,000	53.1	68.7	39.6 – 66.5
10,000	28.9	39.5	21.1 – 36.5
100,000	278.6	16.6	275.3 – 281.9
1,000,000	9473.6	424	9390.54 – 9556.7

As a result of this observation we have modified the algorithm; the new algorithm revisits an already rewired node and continues to iterate until the number of degree-one vertices is close to the theoretical value calculated in Chapter Four for the corresponding value of γ. Then we investigate the stopping criteria for the new algorithm. We have seen in Chapter Three that the number of iterations to reach the value predicted by the theoretical model for the number of vertices of degree k is a function of N and of, the degree of the power law; our experiments confirm this and show that we need about 10^6 iterations of the algorithm when $N = 10^6$.

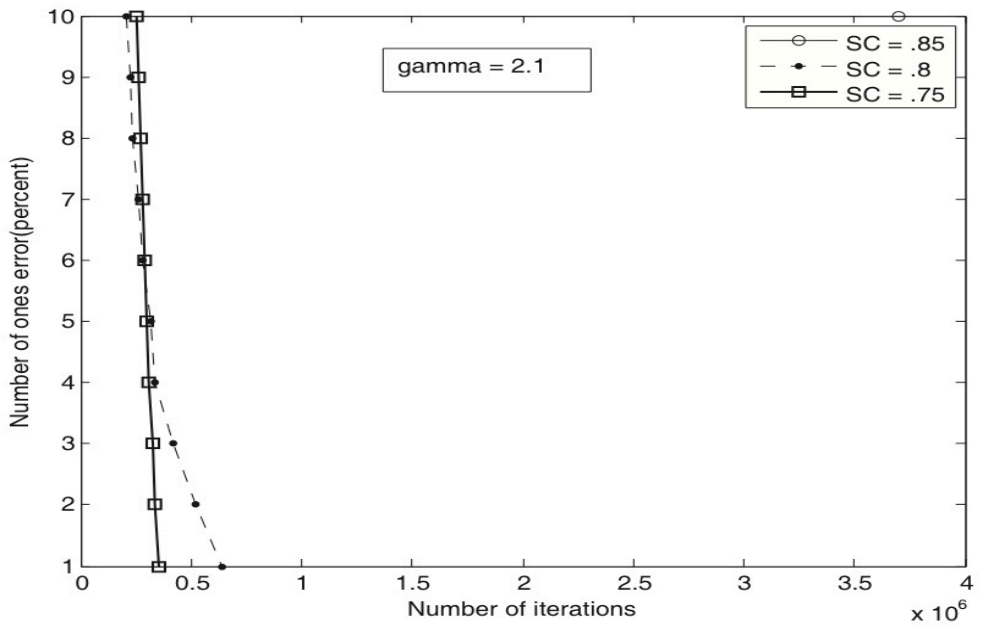
Figure 3 shows that the number of iterations necessary to be within 2% of the predicted value is about 10 × 10^5 when the number of vertices of the random graph increases from N =1,000, to 10,000, and then to 100,000; this number increases by an order of magnitude for N = 1,000,000 vertices. We also notice that the convergence to the theoretical value is slower when the number of vertices of the random graph increases.

We now discuss the timing requirements for the algorithm to create a scale-free overlay network. The execution times required by this version of the algorithm to reach the distribution predicted by the theory are summarized in Table 2. While the time required for the formation of the scale-free network covering an entire cloud with 10^6 computing nodes seems prohibitive, around 2.5 hours, we should keep in mind that the simulations run on an Intel Core 2 Duo E7500 system with 4 GB of memory and a clock rate of 2.93 GHz under Fedora 12 64-bit operating system; in practice, the algorithm will run on much faster systems. The establishment of the global scale-free overlay network is one step of the start-up of the cloud and will run once every few weeks or months. On the other hand, the establishment of the scale-free network for a cloud with less than 1,000 systems will probably take a few seconds in practice.

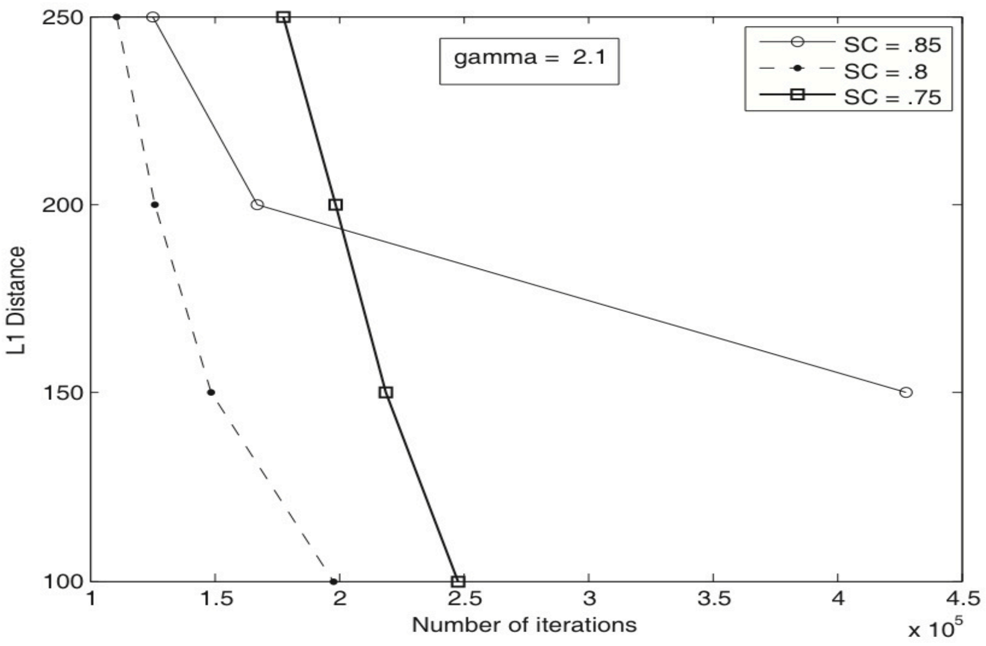
The next question we address regards the error when we compare the number of degree-one vertices with the one predicted by the power-law as a stopping criterion. A more accurate solution is to use the *L1* distance, but this requires collecting information about the degree of all nodes. The distributed algorithm to construct a scale-free network uses the following expression with the constant $k = 1$ to decide the bias used by node i_a to forward the random walk message to a node i_n with d_a and d_b the degrees of vertices i_a and i_n, respectively.

$$\frac{d_a}{d_n}\left(\frac{i_a}{i_n}\right)^{\frac{k}{\gamma\alpha}-1} \qquad (59)$$

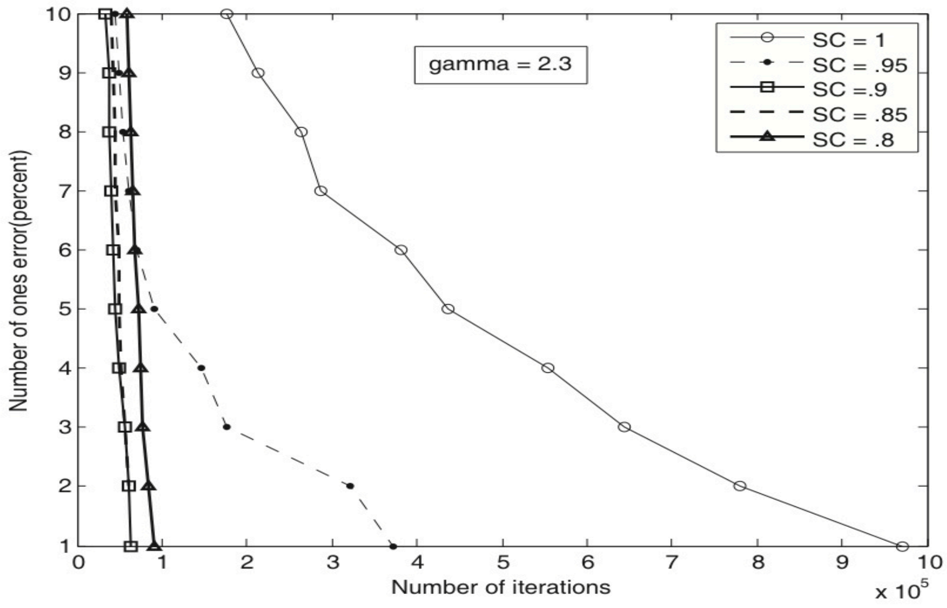
We use as a stopping criteria the *L1* distance between the degree distribution predicted by the theoretical model and that produced by the algorithm; we also compare the number of degree one vertices predicted by the theoretical model and that produced by the algorithm. We observe that the value of k and the number of iterations required to achieve a certain level of error are a function of the power of the degree distribution and, as expected, by the metrics used to compute the error. The results summarized in Figures 4 and 5 are for N = 1000 vertices and represent the averages over 10 runs.



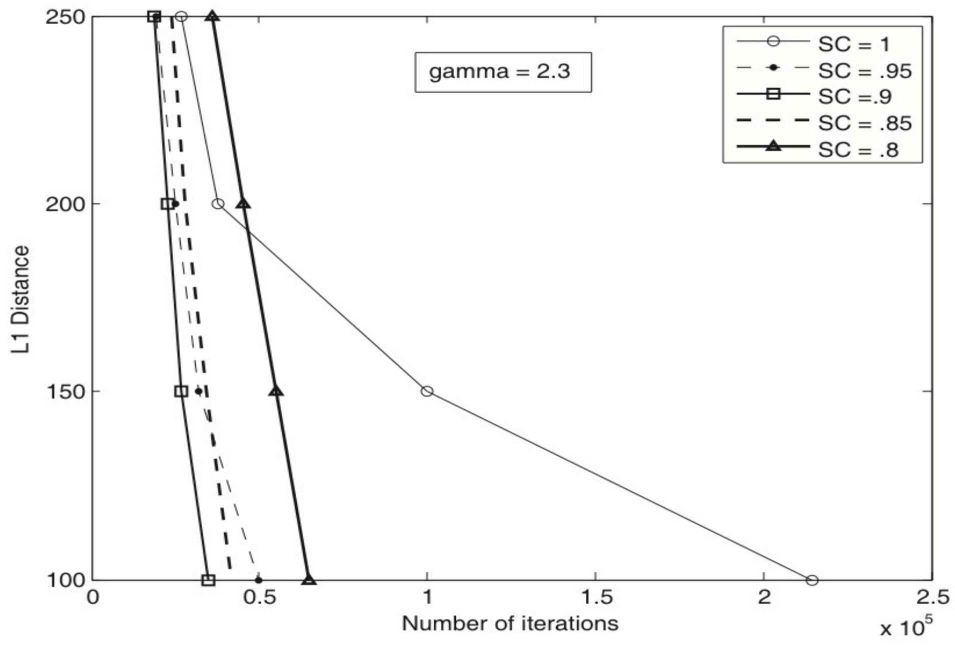
(a)


gamma = 2.1
SC = .85
SC = .8
SC = .75
L1 Distance
250
200
150
100
1
1.5
2
2.5
3
3.5
4
4.5
Number of iterations
x 10^5

(b)


gamma = 2.3
SC = 1
SC = .95
SC =.9
SC = .85
SC = .8
Number of ones error(percent)
10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
Number of iterations
x 10^5

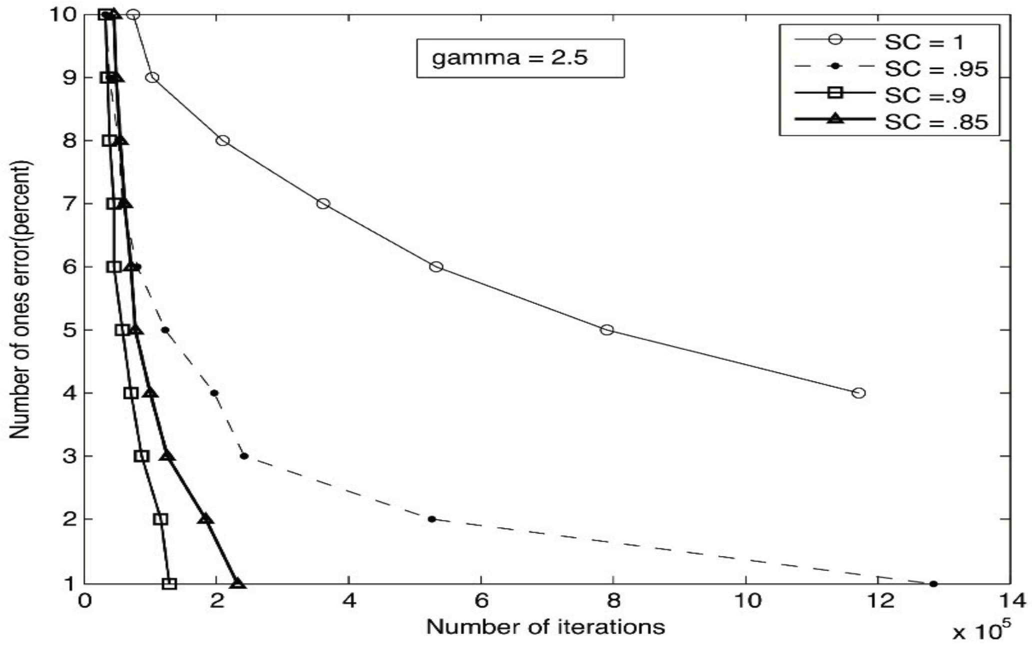
(c)



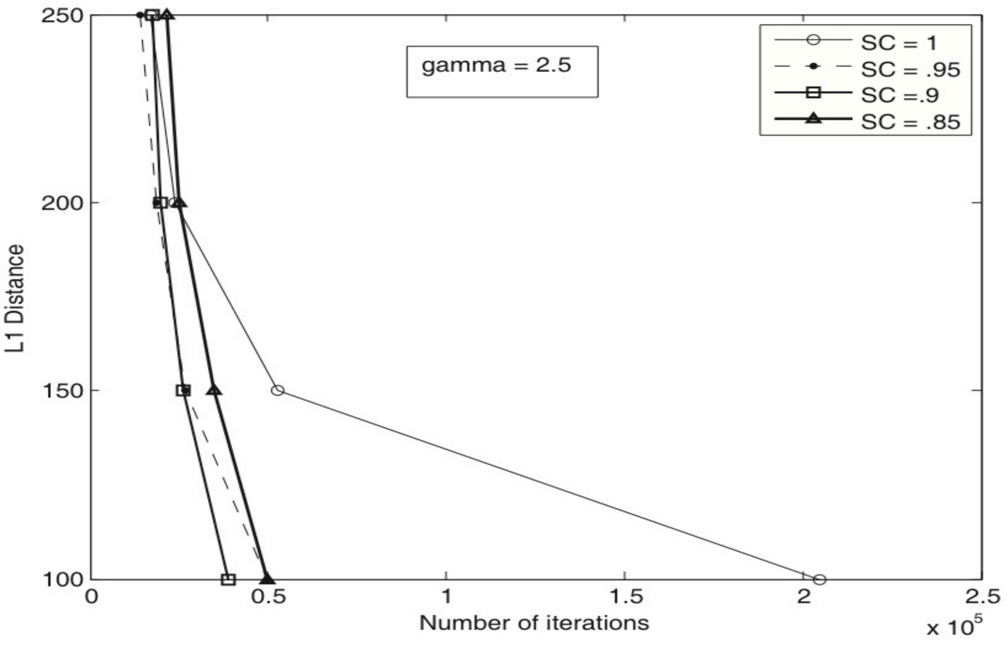
(d)

Figure 4: Simulation Results

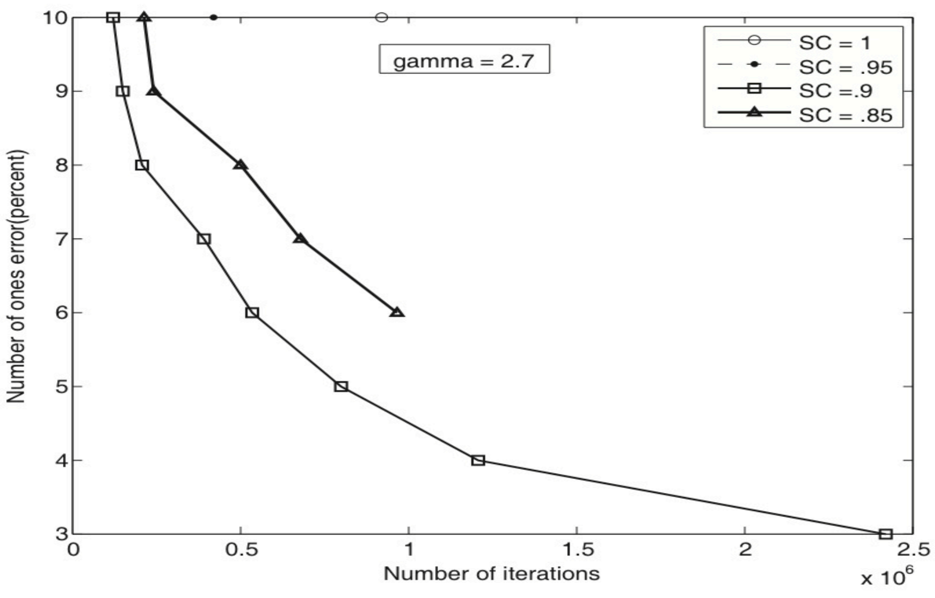
Figure 4(a) shows that when γ = 2.1 the optimal value is *k* = 0:8 and that we need 2×10^5 iterations to reach a distance *L1* equal to 100. If instead of the *L1* norm we use the number of degree-one vertices, as in Figure 4(b), we need around 0.4×10^5 iterations when *k* = 0.75 to be within 1% of the number of degree-one vertices predicted by the theoretical model. Figure 4(c) shows that for γ = 2.3 the optimal value is *k* = 0.9 and we reach a distance *L1* of 100 after about 0.7×10^5 iterations; we need around 0.7×10^5 iterations to be within 1% of the number of degree-one vertices predicted by the theoretical model, Figure 4(d).


gamma = 2.5
SC = 1
SC = .95
SC =.9
SC = .85
Number of ones error(percent)
Number of iterations
x 10^5

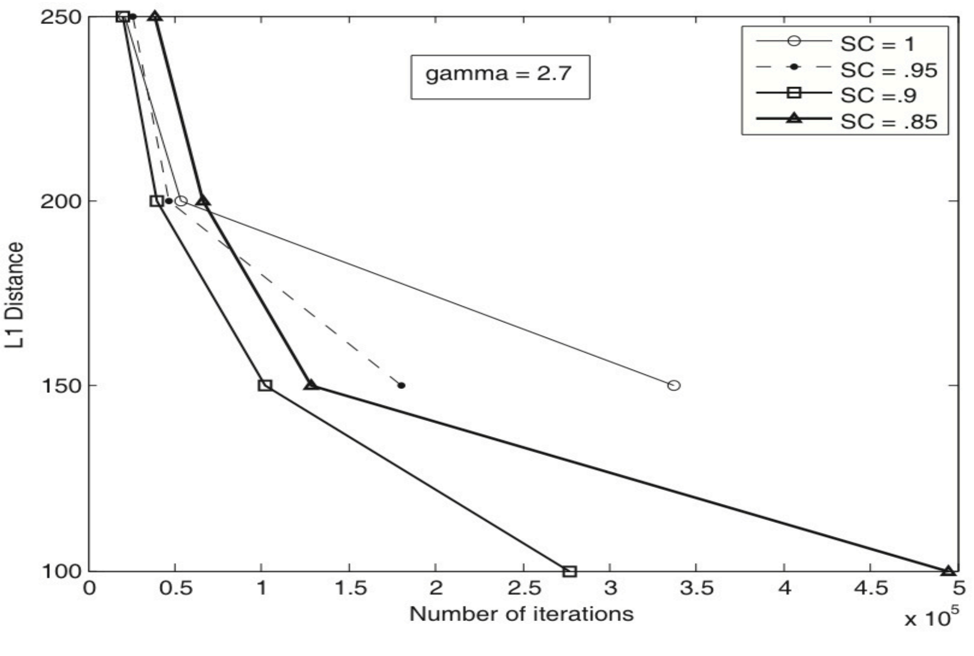
(a)


gamma = 2.5
SC = 1
SC = .95
SC =.9
SC = .85
L1 Distance
Number of iterations
x 10^5

(b)



(c)



(d)

Figure 5: Simulation Results

When $\gamma = 2.5$ the optimal value is $k = 0.9$ and we reach distance *L1* of 100 after about $0 : 4\times10^5$ iterations, shown in Figure 5(a); we need around 1.2×10^5 iterations to be within 1% of the number of degree-one vertices predicted by the theoretical model, shown in Figure 5(b). When $\gamma = 2.7$ the optimal value is $k = 0.9$ and we reach a distance *L1* of 100 after about 2.8×10^5 iterations, shown in Figure 5(c); we need around 2.5×10^5 iterations to be within 3% of the number of degree-one vertices predicted by the theoretical model, shown in Figure 5(d).

Lastly, we report on the creation of a virtual cloud. We started with a scale-free network with $\gamma = 2.5$ and with 10^6 nodes discussed earlier. Then we assembled a scale-free network of 5,000 nodes selected through a random walk from the 330,000 "green computing" nodes of the cloud. The next step was to generate a random number representing free capacities of the 5,000 nodes in this scale-free network. Finally we constructed a normal distribution on 1,000 nodes.

The execution time for building the normal distribution is less than a second and the time for the reconstructing of the SF network for the 5,000 nodes was 155 seconds. Figure 5 shows the distribution of the free capacity of the nodes included in the virtual cloud.

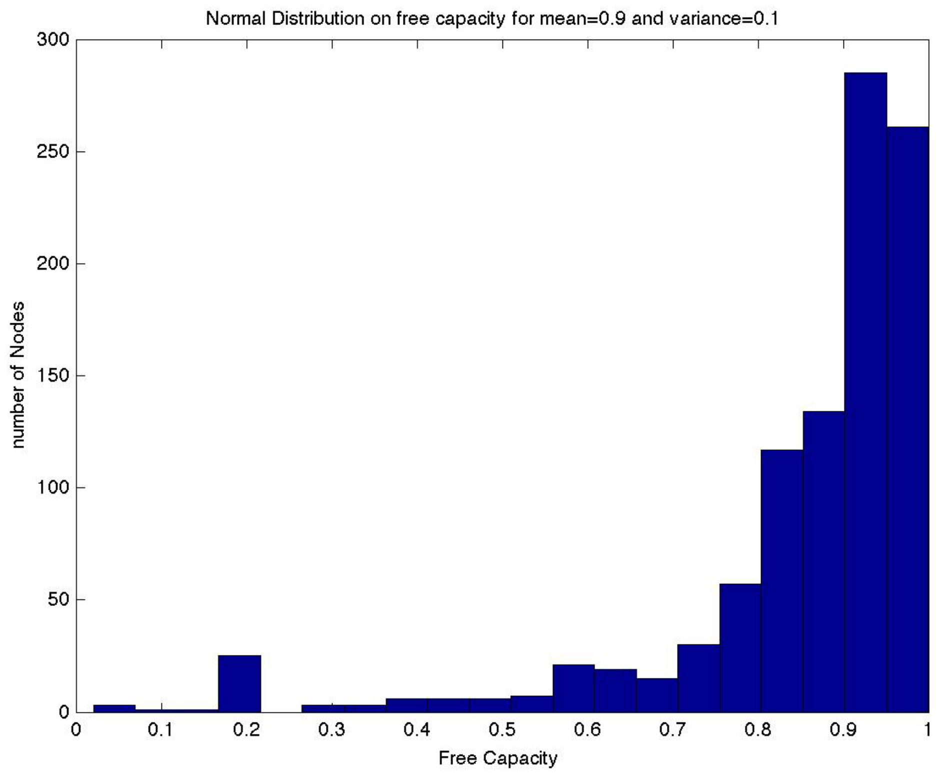


Figure 6: The histogram of the number of nodes in a cloud of 1,000 nodes with a normal distribution of the free capacity, interconnected by a scale-free network of 5,000 nodes. The 5,000 nodes were selected through a random walk in a network of 330,000 of "green-computing" nodes of a cloud with 10^6 nodes.

6: DYNAMIC WORKLOAD DISTRIBUTION

In this section we discuss the ability of a virtual cloud to react promptly to dynamic changes in the workload. We assume that the global overlay network of the cloud, Γ, follows a power law degree distribution with an exponent $2 < \gamma < 3$.

When a core node of Γ accepts the role of the SLA Coordinator and receives a request for an average workload of ω units and a peak workload of $\alpha_{max} \times \omega$ then it determines the number of nodes in the virtual cloud, $N = |C|$. For example, if the average free capacity is k workload units per system, then the number of systems in the virtual cloud could be $N > \left\lceil {}^{\alpha\omega}/_{k} \right\rceil$ with $1 \leq \alpha \leq \alpha_{max}$; the ratio of number of nodes in the overlay network, $M = \lceil \Gamma_C \rceil$ versus the number of those included in the virtual cloud, $\eta = M/N$ can be in the range $5 \geq \eta \geq 10$ and it is determined using algorithms described in [44]. Then the SLA Coordinator initiates the creation of the overlay network Γ_C for virtual cloud C based on its assessment of the global state of the cloud, in particular on the distribution of the free capacity and of the information gathered from the SLA data regarding the workload and the terms of the contract. Once the overlay network is constructed a subset of the nodes are included in the virtual cloud and one of the core nodes of Γ_C included in C is chosen as the *Cloud Supervisor*; its role is to connect the virtual cloud to the SLA Coordinator which communicates directly with the user.

The two parameters α and η ultimately control M and N, the number of nodes in Γ_C and C, respectively, are affected by the contractual obligations and the penalties. An SLA specifying stiff penalties could be accommodated by *Overprovisioning*; it requires a choice of α closer to

α_{max} . To avoid *Overprovisioning* we could use a larger pool of potential sites for the distribution of the overload, in other words to select a moderate value of α and a large value of η.

Several workload allocation strategies are possible: increase uniformly the load of nodes visited during the random walk; a greedy strategy is to saturate each node visited during the random walk. We say that a node is *saturated* if its load is 90% of its capacity. In the first case the load allocated to the node is a small fraction of its free capacity; this strategy allows multiple virtual clouds operating on the same node to accommodate overloads, but requires a larger number of steps for the random walk. The greedy strategy reduces the number of steps of the random walk; when the cloud is lightly loaded this strategy together with a proper choice of the distribution of the free capacity used to select the nodes of C allows a limited number of nodes to accommodate the entire load of the cluster and turn off the other nodes to save power.

Effective strategies for dynamic load management allow virtual clouds to respond to global surges, an individual surge, or the failures of one or more nodes. A global surge could be triggered by a sudden increase in demand due to a catastrophic event e.g., the blackout in some region of the country, while an individual surge is possible due to the large peak to average ratio of resources specified by a single user. First, we attempt to distribute the additional workload to the N systems in the C; then, if necessary, the residual workload is distributed to the M - N other systems connected by Γ_C, the virtual cloud's overlay network; as a last resort we extend the random walk to nodes outside Γ_C.

The ability of the virtual cloud to accommodate a surge depends on the current load; we consider three scenarios: a lightly loaded, a medium, and a heavily loaded virtual cloud. The average load in these three cases is 10%, 50% and 80%, respectively, of the capacity of the systems included in the virtual cloud.

Next we attempt to quantify the attributes of the virtual cloud important for dynamic workload management. The average number of steps of the random walk measures the time to locate the systems able to carry the extra load; the desired versus the actual distribution of the free capacity after applying an allocation strategy for the extra workload is another useful measure that reflects in part the properties of the overlay network. We also want to see if these measures scale with the size of the virtual cloud $|C|$ and of the overlay network $|\Gamma_C|$.

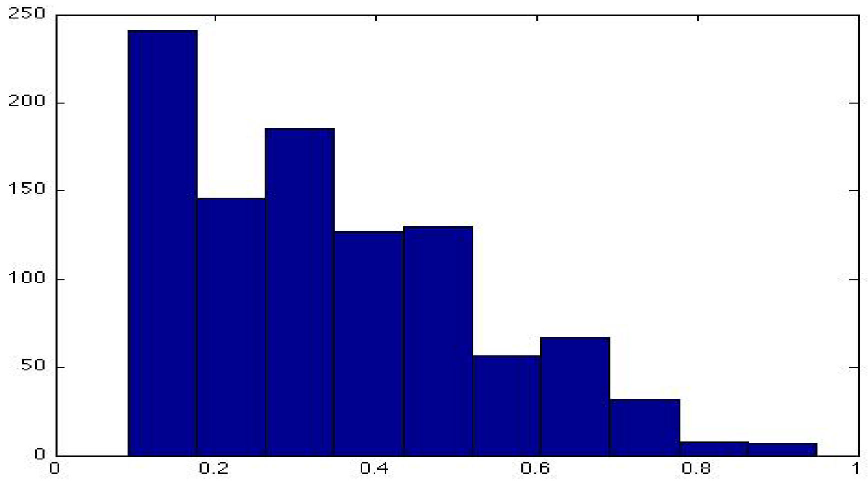
Table 3 shows that a biased random walk in an overlay network with a power law degree distribution requires about the same number of steps as a biased random walk in a random network provided that we do not require to revisit the source node of the random walk when the condition is not met.

Table 3: The number of biased random walk steps for the creation of a virtual cloud in a lightly loaded system when N = 1,000 and M = 5,000. Four experiments are conducted: I – random walk in a random overlay network; II - random walk in an overlay network with a power law degree distribution without the condition revisit the source node of the random walk when the condition is not met; III - random walk in a graph with a power law degree distribution when the random walk is forced to revisit the source node when the condition is not met; IV - workload distribution in an overlay network with a power law degree distribution when we visit all the neighbors of a core node, then repeat the same process for the next core node.

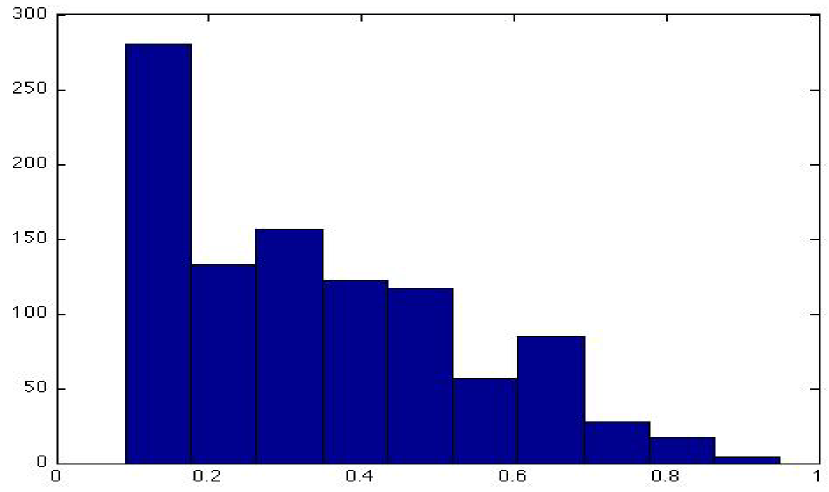
Experiment	Number of steps
I	11,230
II	11,740
III	14,320
IV	43,227

It also shows that a biased random walk in an overlay network with a power law degree distribution requires about three times less steps for workload distribution (11,740 versus 43,227) than when we visit all neighbors of a core node then move to the next core node of the overlay network $|\Gamma_C|$ of the virtual cloud C.

In our simulation experiments a global surge is defined as an increase of the total load of all the N systems included in the virtual cluster by a factor of 10. The virtual cloud consists of 1,000 systems selected through a random walk out of 5,000 systems. For simplicity, we assume that the average load of a system in C is μ_c; then the global surge amounts to $10 \times \mu_c$ workload units. A simple calculation shows that such a global surge cannot be accommodated by a virtual cloud with a heavy load thus; the first group of experiments covers only light and medium loaded virtual clusters.

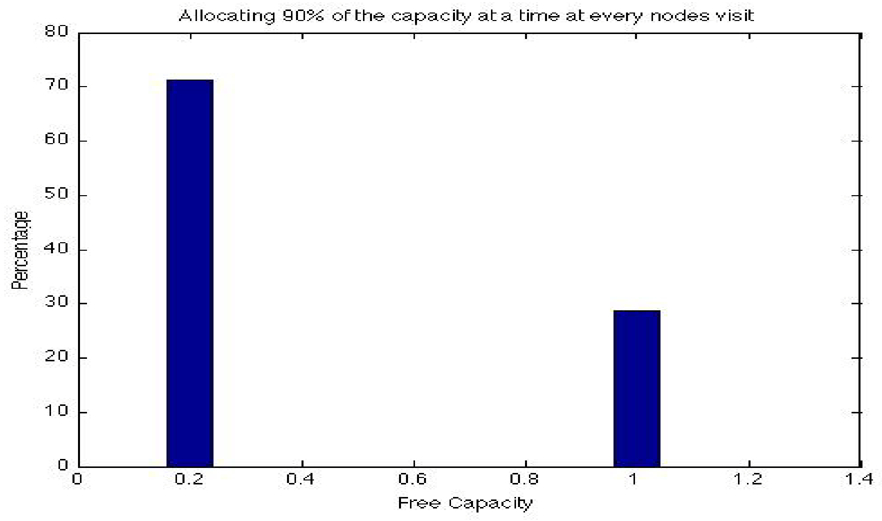


(a)

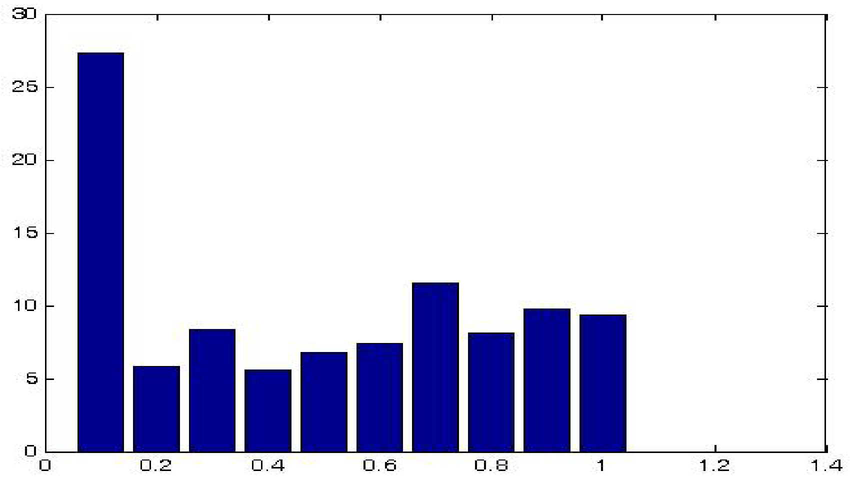


(b)

Figure 7: The distribution of the free capacity in a lightly loaded virtual cloud after a global surge. The virtual cloud is interconnected by the overlay network modeled by: (a) a random graph; (b) a graph with a power law degree distribution.



(a)



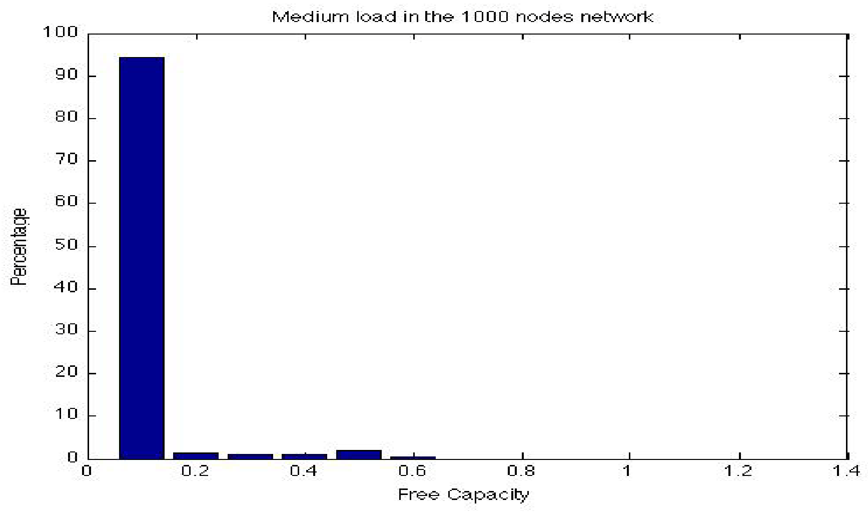
(b)

Figure 8: The distribution of the free capacity in a lightly loaded virtual cloud after a global surge for an overly network with a power law degree distribution. (a) Greedy allocation for the N = 1,000 nodes of the virtual cloud C. (b) Uniform allocation of 10% of the free capacity of individual nodes extends the random walk to the M = 5,000 nodes of $|\Gamma_C|$.

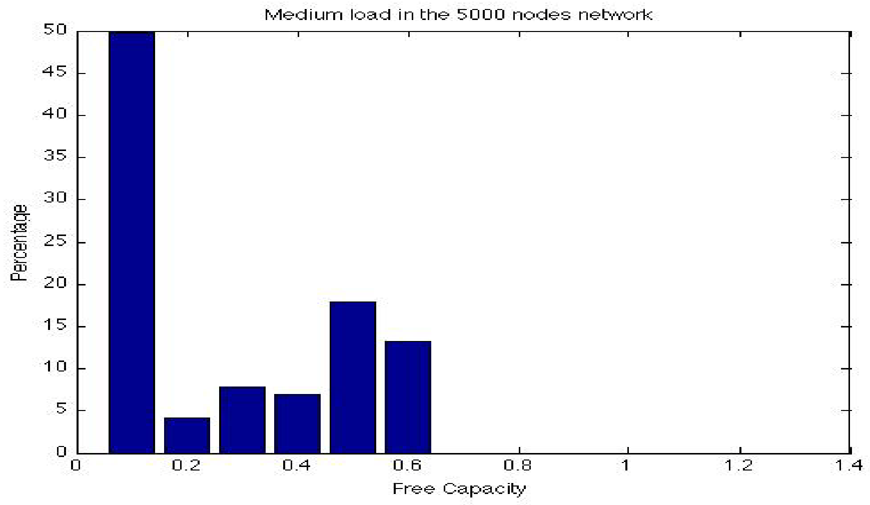
First, we compare the effects of a random walk on overlay networks with different topology for a global surge and a uniform allocation of the extra load described earlier on a lightly loaded virtual cluster. The distribution of the free capacity after the surge for an overlay network modeled as a random graph is shown in Figure 7 (a) and the one for a power-law degree distribution is shown in Figure 7 (b). The distribution of the free capacity as well as the number of random walk steps are very similar; for the random graph some 24% of the virtual cluster nodes are saturated (loaded up to 90% of their capacity) versus 28% for the power-law degree distribution.

The next sets of experiments consider only a power-law degree distribution of the overlay network and we compare the greedy with the uniformly load allocation. Figures 8 (a) and (b) show the distribution of the free capacity in a lightly loaded virtual cloud while Figures 9 (a) and

(b) show the distribution of the free capacity in a medium loaded virtual cloud after a global surge.



(a)

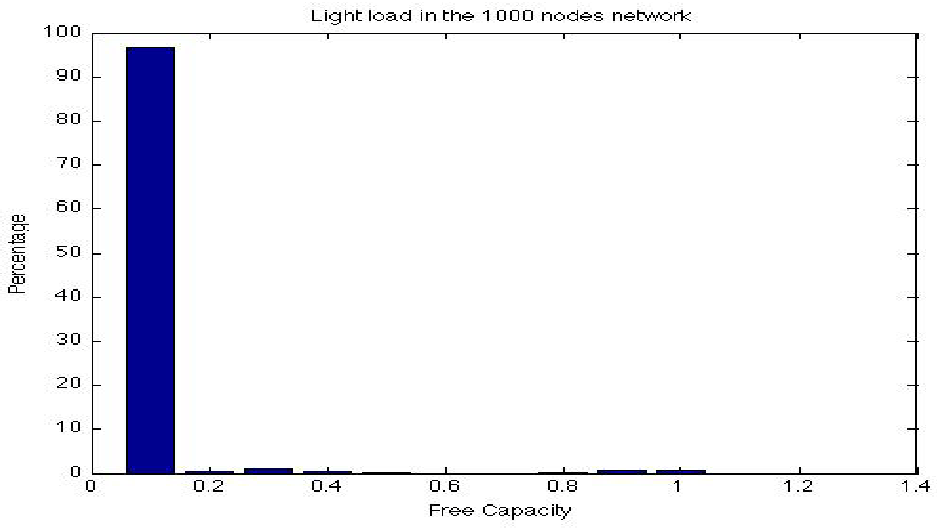


(b)

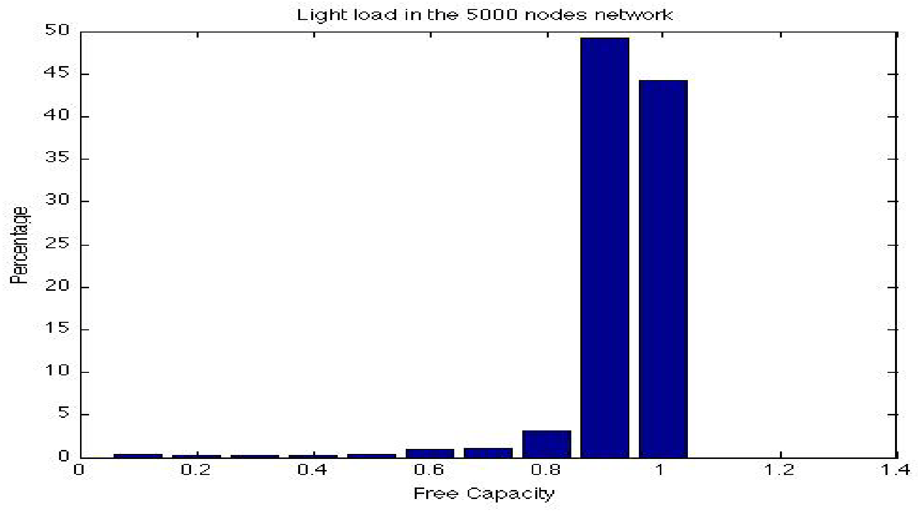
Figure 9: The distribution of the free capacity in a virtual cloud with a medium load after a global surge for an overly network with a power law degree distribution. (a) Greedy allocation on the N = 1,000 nodes of the virtual cloud C. (b) Uniform allocation of 10% of the free capacity of individual nodes extends the random walk to the M = 5,000 nodes of $|\Gamma_C|$.

In a lightly loaded system a greedy allocation saturates about 70% all the N = 1,000 nodes of C as we can see Figure 8 (a). Figure 8 (b) shows that when we increase the load on each node by only 10% we have to extend our search to the M = 5,000 nodes in $|\Gamma_C|$ and about 27% of them are saturated.

In a medium loaded system a greedy allocation saturates practically all the N = 1,000 nodes of C as we can see in Figure 9 (a). On the other hand, Figure 9 (b) shows that when we increase the load on each node by only 10% we have to extend our search to the M = 5,000 nodes in $|\Gamma_C|$; then about 50% of them (about 2,500 are saturated. The random walk covered 85% of the nodes of $|\Gamma_C|$.

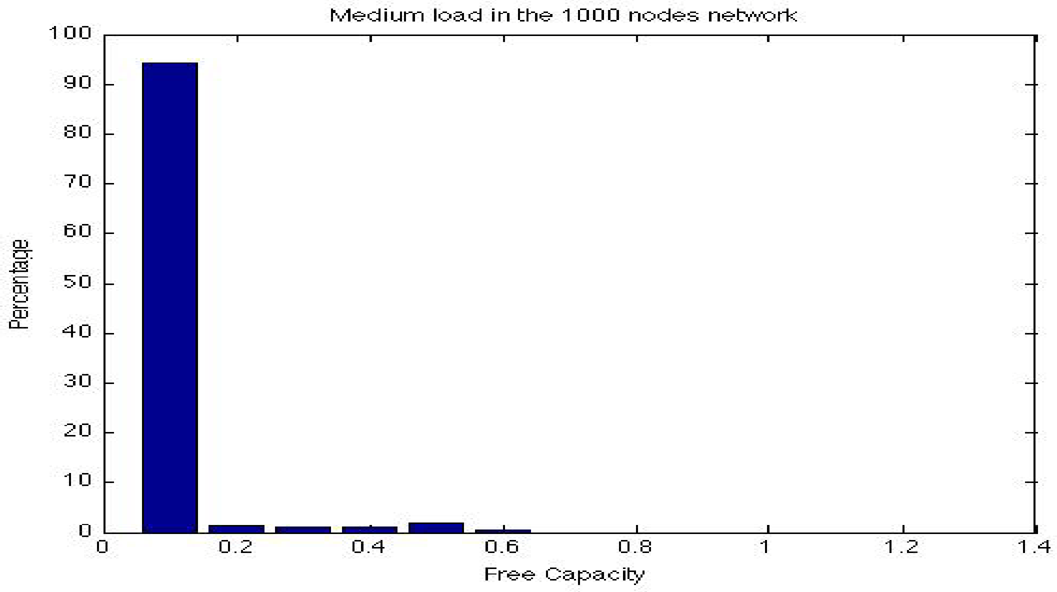


(a)

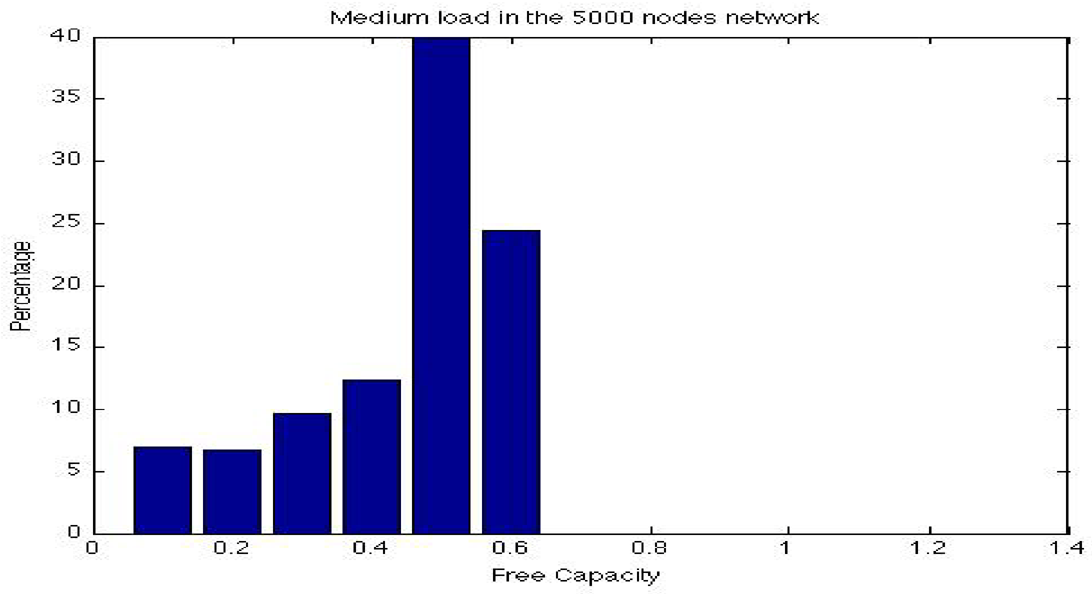


(b)

Figure 10: A surge in a system with the average free capacity 10% of the total capacity.



(a)

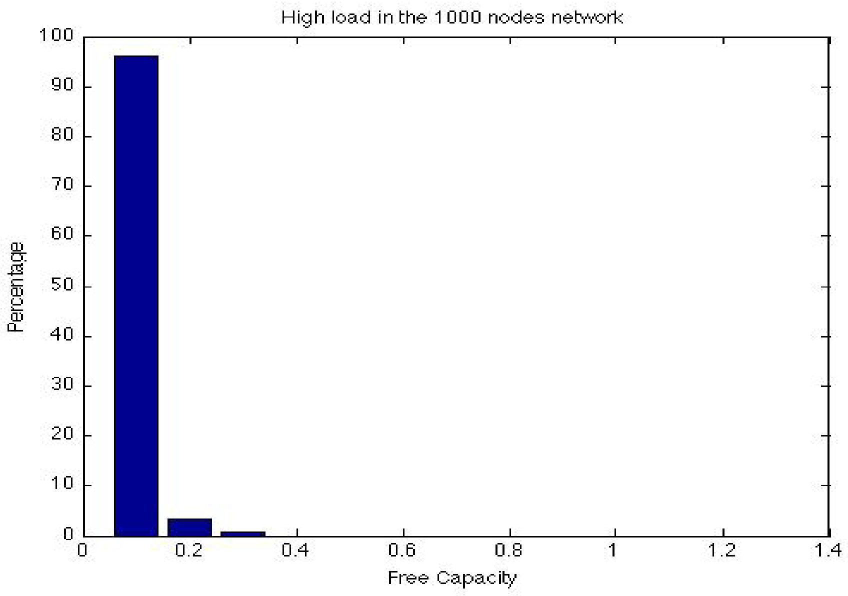


(b)

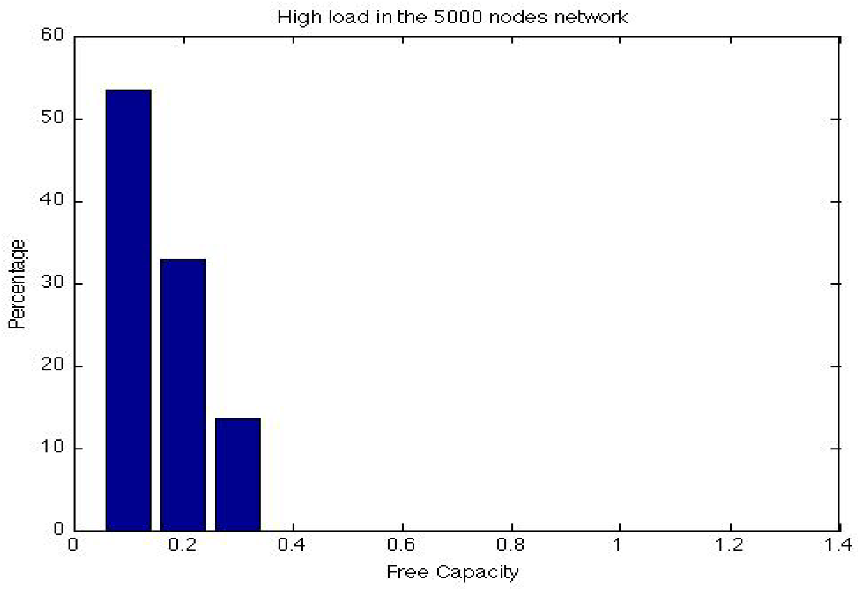
Figure 11: The distribution of the free capacity in a virtual cloud with a medium load after a local surge for an overly network with a power law degree distribution. (a) Greedy allocation on the N = 1,000 nodes of the virtual cloud C. (b) Uniform allocation of 10% of the free capacity of individual nodes extends the random walk to the M = 5,000 nodes of $|\Gamma_C|$.

Next we consider local surges due to a 10 fold increase of the workload of cluster C and examine only the medium and high load cases illustrated in Figures 11 (a) and (b) and in Figures 12(a) and (b), respectively. In each case we show the distribution of the free capacity for a greedy and for a uniform distribution of the load.

In case of a virtual cloud with a medium load the greedy allocation leads to the saturation of 95% of the nodes of C, as shown in Figure 11 (a). When we increase the load on each node by only 10% we have to extend our random walk to the M = 5,000 nodes in $|\Gamma_C|$ and we require 240 steps to accommodate the extra workload; Figure 11 (b) shows that in this case only 6% of the nodes in $|\Gamma_C|$ are saturated.



(a)



(b)

Figure 12: The distribution of the free capacity in a heavily loaded virtual cloud after a local surge for an overly network with a power-law degree distribution. (a) Greedy allocation on the N = 1,000 nodes of the virtual cloud C. (b) Uniform allocation of 10% of the free capacity of individual nodes extends the random walk to the M = 5,000 nodes of $|\Gamma_C|$.

Figure 12 (a) shows similar results when the virtual cloud is heavily loaded and a greedy allocation strategy is in effect; 95% of the nodes of C. When we increase the load on each node by only 10% we have to extend our random walk to the M = 5,000 nodes in $|\mathit{\Gamma}_C|$, but this time we need 2,200 additional steps; Figure 12 (b) shows that in this case only 54% of the nodes in $|\mathit{\Gamma}_C|$ are saturated. These results are consistent with our intuition and with the results for the medium load.

7: SUMMARY AND FUTURE WORK

The work reported in this thesis is not restricted to a specific computing, communication, or storage model and it is complementary to the research on virtualization carried out now by several groups from industry [22], [23]. The virtualization architecture we propose is dynamic, virtual machines are created in response to an external request when local condition permit and have a limited lifetime. A virtual machine created on a node must act in concert with the other members of the virtual cloud and maintain only limited information about its neighbors in the overlay network. The virtual cloud created in response to a SLA provides the services at a minimal cost and with minimal energy consumption. We propose a probabilistic approach for resource virtualization in a computer cloud based on biased random walks; the algorithm allows us to select a subset of systems and to create the overlay network interconnecting these systems.

We discuss the algorithms to carry out the random walks when the transition matrix is *d*-dimensional as the systems included in the virtual cloud are interconnected by a scale-free overlay network; these systems are selected through a random walk and could be subject to additional constrains such as limits on the cost of per unit of service, total cost, or the requirement to use only “green” computing cycles. Scalability is an obvious, but often ignored requirement in system design; typically, scalability becomes an issue only if a system is successful. Attributes such as functionality, reliability, security, and cost are the only concerns in the early stages of system development. The view expressed in this paper is that scalability should be an ab-initio concern; the logical organization or the overlay interconnection network is expected to follow a power-law distribution. The self-management scheme we propose takes advantage of the remarkable properties of scale-free networks such as robustness against random failures, favorable scaling, and resilience to congestion, small diameter, and average path length.

Our preliminary results reported in Chapter Six show that the algorithms we propose for the creation on virtual clouds are relatively efficient and can be used for the implementation of a test bed system. We plan to develop parallel versions of the algorithm for the random walk and expect a substantial reduction of the time to create a scale-free network with $10^6 - 10^8$ vertices.

We collaborate with a group from the University College Cork in Ireland on a physical implementation of a test bed system based on the WebCom infrastructure [41]. The implementation of the virtual clouds will take advantage of existing software systems for resource management such as the capacity planner discussed in [43], the Xin credit scheduler [49], and performance analysis tools such as the Tivoli system developed by IBM [28].

The software architecture we work on for the individual systems in a computing cloud includes a top level Virtual Clouds Manager (VCM) which supervises multiple Virtual Cloud Engines (VCE) each one of them created in response to an SLA. The decision to join a virtual cloud is based on local information provided by the capacity planner and by the power management system running on each system; once a bid to join a virtual cloud is accepted, the VCM creates the virtual cloud engine for the SLA. A VCE is a virtual machine responsible for communications with other members of the virtual cloud and for the execution of the sub-set of activities of the SLA assumed when the local system joined the virtual cloud.

A service-level agreement spells out QoS guarantees as well as elements required to determine the Classes of Service for different activities required by the SLA; given the wide range of service demands the VCE should have access to an ontology to track various aspects of performance monitoring of the SLA [19]. The virtual cloud engine gathers information about its neighbors in the overlay network, then parses the SLA to identify the set of events to be tracked during the lifetime of the contract as well as the actions required by each event.

REFERENCES

[1] R. Albert, H. Jeong, and A.-L. Barabasi. "The diameter of the world wide web." *Nature*, **401**:130, 1999.

[2] R. Albert, H. Jeong, and A.-L. Barabasi. "Error and attack tolerance of complex networks." *Nature*, **406**:378382, 2000.

[3] R. Albert and A-L. Barab´asi. "Statistical mechanics of complex networks." *Reviews of Modern Physics*, **72**(1):48–97, 2002.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, r. Katz,, A konwinski, G. Lee, D. Paterson, A. Rabkin, I. Stoica, M. Zaharia. "Above the clouds: a Berkeley view of cloud computing." *Technical Report UCB/EECS-2009-28.* 2009.

[5] M.J. Atallah, D. C. Marinescu, C.L. Black, H.J. Siegel, and T.L. Casavant. "Models and algorithms for co-scheduling compute-intensive tasks on a network of workstations." *Journal of Parallel and Distributed Computing (JPDC)*, **16**(4):319–327, 1992.

[6] A. A. Baker. "Monte Carlo simulations of radial distribution functions for a proton-electron plasma." *Aust. J. Phys*. **18**:119-133, 1965.

[7] A-L Barab´asi and R. Albert. "Emergence of scaling in random networks." *Science*, **286**:509–512, 1999.

[8] A-L. Barab´asi, R. Albert, and H. Jeong. "Scale-free theory of random networks; the topology of world wide web." *Physica A*, **281**:69–77, 2000.

[9] B. Bollobas. *Random graphs*, Academic Press, London, 1985.

[10] Bonabeau, E., Dorigo, M., and Theraulaz, G. "Inspiration from optimization from social insect behavior." *Nature 406*, 39–42, 2000.

[11].N. F. Britton. *Essential Mathematical Biology*. Springer Verlag, 2004.

[12] Camazine, S. F., Deneubourg, J.-L., Franks, N. R., Sneyd, j., Theraulaz, G., and Bonabeau, E. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ., 2001.

[13] A. Clauset, C. R. Shalizi, and M. E. J. Newman. "Power-law distributions in empirical data." *Siam Reviews*, **51**:661-704, 2007.

[14] R. Cohen and S. Havlin. "Scale-free networks are ultrasmall." *Phys. Rev. Lett.*, **90**(5):058701, 2003.

[15] Collier, T. C. and Taylor, C. “Self-organization in sensor networks.” *Journal of Parallel and Distributed Computing (JPDC),* **64**(7):866–873, 2004.

[16] T M. Cover and J. A. Thomas. “Elements of information theory,” Wiley, New York, NY, 1991.

[17] S. N. Dorogovtsev and j. F. F. Mendes. “Evolution of networks with aging of sites.” Phys. Rev. E, **62**(2):18421845, 2000.

[18] P. ErdÖs and A. Renyi. “On random graphs.” *Publicationes Mathematicae* **6**: 290297, 1959.

[19] A. D. H. Farwell, M. J. Sergot, M. Salle, C. Bartolini, D. Tresour, A. Christodoulou. “Performance monitoring of service-level agreements for utility computin.” *Proc IEEE. Int. Workshop on Electronic Contracting (wec04),* 2004.

[20] S. Garfinkel. “An evaluation of Amazon’s grid computing services: EC2, S3, and SQS.“ *Technical Report*, tr-08-07, Harvard University, 2007.

[21] C. Gkantsidis, M. Mihail, a. Saberi. “Random walks in peer-to-peer networks." *Performance Evaluation*, **63**(3): 241{263, 2006.

[22] D. Gmach, S. Kompass, A. Scholz, M. Wimmer, and A. Kemper. “Adaptive quality of service management for entreprize services.” *ACM Trans. on the Web (TWEB)* **2**(1):243–253, 2009.

[23] D. Gmach, J. Rolia, and L. Cerkasova. “Satisfying service-level objectives in a self-managed resource pool.” *Proc. 3rd. Int. Conf. On Self-Adaptive and Self-Organizing Systems*. pp. 243–253, 2009.

[24] K. I Goh, B. Kahang, and D. Kim. “Universal behavior of load distribution in scale-free networks." *Physical Review Letters*, **87**:278701, 2001.

[25] I. Gupta, A J. Ganesh, A-M kermarrec. “Efficient and adaptive epidemic-style protocols for reliable and scalable multicast.” *IEEE Trans. on Parallel and Distributed Systems.* ***17(7)***:593-605, 2006.

[26] W. K. Hastings. “Monte Carlo sampling methods using markov chains and their applications." *Biometrika*, **57**:97109, 1970.

[27] Hopfield, J. “Neural networks and physical systems with emergent collective computational abilities.“ *Proc. National Academy of Science 79*, 2554–2558, 1982.

[28] IBM. “Tivoli performance analyzer.” www.ibm.com/software/tivoli/products/performance-analyzer, 2008.

[29] M. Jelasity, A. Montresor, and O. Babaoglu. “Gossip-based aggregation in large dynamic networks.” *ACM Transactions on Computer Systems*, **23**(3):219252, 2005.

[30] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen. “Gossip-based peer sampling.” *ACM Trans. Comput. Syst.*, **25**(3):8, 2007.

[31] W. O. Kermack and A. G. Mckendrick. “ A contribution to the theory of epidemics.” *Proc. Royal Soc. London, A*, **115**:700–721, 1927.

[32] A. N. Kolmogorov. “Three approaches to the quantitative definition of information.” *Problemy Peredachy Informatzii*, **1**:4-7, 1965.

[33] Krugman, P. R. *The self-organizing economy*. Blackwell Publishers, 1996.

[34] D. S. Lee, K. I. Goh, B. Kahng, and D. Kim. “Evolution of scale-free random graphs: potts model formulation.” *Nuclear Physics B*. **696**:351– 380, 2004.

[35] D. C. Marinescu, J. P. Morrison, and H. J. Siegel. “Options and commodity markets for computing resources.” In *market oriented grid and utility computing*, R. Buyya and K. Bubendorf, Eds., Wiley, 2009.

[36] D. C. Marinescu, c. Yu, and G. M. Marinescu. “Self-organization of very large sensor networks based on small-worlds principles.” *Proc. Third IEEE Conf. On self-adaptive and Self-Organizing Systems, SASO- 09*, pp.115-125, 2009.

[37] D. C. Marinescu, C. Yu, and G. M. Marinescu. “Scale-free, self-organizing very large sensor networks.” *Journal of Parallel and Distributed Computing (JPDC)*, **50**(5):612-622, 2010.

[38] von der Marlsburg, C. “Network self-organization.“ *In an Introduction to Neural and Electronic Networks*. S. Zonetzer, J. L. Davis, and C.Lau (Eds.), 421-432, Academic Press, San Diego, CA, 1995.

[39] N. Metropolis, A. W. Rosenbluth, A. Teller, and E.teller. “Equation of state calculations by fast computing machines." *J. of Chemical Physics*, **21**(6):1097{1092, 1953.

[40] A. Mondal, S. K. Madria, and M. Kitsuregawa. “Abide: a bid-based economic incentive model for enticing non-cooperative peers in mobile p2p networks," *Proc. Database Systems for Advanced Applications, DAS-FAA* 703{714, 2007.

[41] J P. Morrison, B. Clayton, D. A. Power and A. Patil. “WebCom-G: grid enabled metacomputing,” *The Journal of Neural, Parallel and Scientific Computation*, Special Issue on Grid Computing. H.R. Arabnia, G.S. Gravvanis, m.P. Bekakos, Eds. Vol. **12**(3):419–438, 2004.

[42] M. E. J. Newman. “The structure of scientific collaboration networks.”*Proc. Nat. Academy of Science*, **98**(2):404–409, 2001.

[43] J. Rolia, L. Cerkasova, M. Arlit, and A. Andrzejak. “A capacity management service for resource pools.” *Proc. 2nd Symp. on Software and Performance*. pp. 224–237, 2005.

[44] M. Saleh and Dan C. Marinescu. *Self-organization and virtualization for service-level agreements on computing clouds*, 2011, (submitted).

[45] I. Sholtes, J. Botev, A. Hohfeld, and H. Schloss. “Awareness-driven phase transitions in very large scale distributed systems,” *Proc. SASO-08, Second IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems, IEEE Press*, pp. 25–34, 2008.

[46] I. Scholtes. “ Distributed creation and adaptation of random scale-free overlay networks.” *Proc. Fourth IEEE Int. Conf. of Self-Adaptive and Self-Organizing Systems, SASO-10*, 2010 (in print).

[47] Z. Toroczkai and K. E. Bassler. “Jamming is limited in scale-free systems.” *Nature,* **428**:716, 2004.

[48] A. M. Turing. “The chemical basis of morphogenesis,“ *Philos. Trans. Roy. Soc. London b*,**237**: 37-72, 1952.

[49] Xen wiki. http://wiki.xensource.com/xenwiki/creditscheduler, 2007.

[50] D. J. Watts and S. H. Strogatz. “Collective-dynamics of small-world networks,” *Nature,* **393**:440–442, 1998.