A power-law graph as a distributed hash table with quick search and small tables

Hannu Reittu, and Ilkka Norros
VTT Technical Research Center of Finland
P.O. Box 1000, 02044 VTT
Finland
{ilkka.norros, hannu.reittu}@vtt.fi

Abstract*—We analyze the possibility of using an 'Internet-like' power-law graph as a basis for peer-to-peer distributed hash table applications. Our work is based on previous studies of power-law random graph models that showed emergence of a spontaneous hierarchy of nodes based on their degrees, called the 'soft hierarchy'. The soft hierarchy indicates very short paths, leading to the top of the hierarchy, where the top consists of a clique of highest degree nodes. Such paths have lengths that scale as $\log\log N$, with number of nodes N. Further, such paths can be found by a heuristic rule: 'the next hop to highest degree neighbor'. We suggest that these circumstances could be used as a basis of an efficient distributed hash table. The idea is that the hash-entries, needed to locate content, are stored at the periphery of the hierarchy, consisting of large enough set of nodes to guarantee small tables. It is required that any node, say, i in the hierarchy is aware which nodes are below it in the hierarchy, provided it is not in the periphery. The node i places its 'down-hill' neighbors in the hash-ring with equal intervals between them. When node i gets a request to store or search a given hash-entry, it uses some locally defined function that places the hash value on this ring and forwards it to the down-hill neighbor closest to this value. Our main result is a probabilistic estimate of the number of hash-values stored in a periphery node. It appears to be sub $\log\log N$ and super $\log\log\log N$, with $N \to \infty$, and with probability tending to 1. Another contribution is a sketch of a novel algorithm that would create such topologies in a self-organizing manner.

I. Introduction

We consider here one case of such a hybrid approach. We study whether a so called 'power-law graph', intensively studied in connection to Internet and other complex networks, could serve as a basis for such networks. In previous studies of a random variant of such graphs, it was found that the large degree nodes play an important role in the connectivity of nodes with short paths that scale as $\log\log N$ with the number of nodes in the 'giant component' (the largest [in number of nodes] connected component of the graph). We suggest that such large nodes present the provider side of the network. They are also assumed to be more server-like in the sense of non-volatility and larger capacity. The rest of the network, the majority of nodes, is the peer side of the network, constituted from voluntaries. We show that such a network can be made self-organizing in a way that it gradually fits the demands while the network becomes large. We show also that in a power-law random graph it is possible to establish a distributed hash table scheme that is very efficient in terms of search path lengths and the hash table sizes. The first one scales as $\log\log N$ [1], [2], [3] and the last one even better than that. Although the table sizes grow asymptotically to infinity, the rate is extremely slow, sub $\log\log N$ and super $\log\log\log N$. In the best possible case we would have constant size tables. Thus we have some penalty in table sizes, but hardly noticeable in practice. The power-law graph scheme is based on our previous work on a spontaneous hierarchy of high degree nodes that form a 'core' of the random power-law graph. The results hold with probability tending to 1 as $N \to \infty$, or in short, *a.a.s.*.

A benefit of such an approach for the provider side could be relaxed burden on infrastructure, since the peers also participate and the network is self-organizing. The provider has also up to date track on demands. The peers would benefit from a quick search and high capacity system without high cost overhead of heavy infrastructure.

VALUETOOLS 2009, October 20-22, Pisa, Italy
Copyright © 2009 ICST 978-963-9799-70-7
DOI 10.4108/ICST.VALUETOOLS2009.7724

II. The conditionally Poissonian IVPLRG model

Consider graphs with a set V of vertices (henceforth, 'nodes') and a set E of non-directed edges (henceforth, 'links'). The number of nodes is denoted by $N := |V|$. Two nodes connected by a link are called neighbors. The degree of a node is the number of its neighbors.

The conditionally Poissonian random graph [4] is constructed as follows. First, each node is given a 'capacity' Λ_i with a power-tail distribution. For simplicity, assume that the capacities obey the Pareto distribution

$$\mathbb{P}(\Lambda > x) = x^{-\alpha}, \quad x \in [1, \infty), \quad \alpha \in (1, 2). \tag{1}$$

The choice of the parameter α is crucial: the graph has entirely different properties if α is taken outside of $(1, 2)$, where Λ has finite mean but infinite variance. In the second step, each pair of nodes $\{i, j\}$ is independently joined by E_{ij} links, where E_{ij} is a Poisson random variable with mean

$$\frac{\Lambda_i \Lambda_j}{L_N}, \quad L_N := \sum \Lambda_k. \tag{2}$$

Loops and parallel links are accepted but mostly neglected. Note that (2) is a 'gravity rule': the expected number of links between nodes i and j is proportional to both capacities Λ_i and Λ_j. Conditioned on the capacities, the degree of node i is a random variable with distribution Poisson(Λ_i). By the addition rule of Poissonian random variables, the number of edges joining two disjoint sets of nodes is again Poissonian. From now on, we use the term IVPLRG for this particular model. Rigorous proofs of the claims below can be found in [4], [5], and a more intuitive discussion in [6]. Figure 1 helps with the presentation.

A. *The core network and its soft hierarchy*

The core network: Choose $\epsilon(N)$ satisfying the following three conditions as $N \to \infty$:

$$\epsilon(N) \to 0, \quad N^{\epsilon(N)} \to \infty, \quad \frac{N^{\epsilon(N)}}{\log \log N} \to 0. \tag{3}$$

We define the *core* of the IVPLRG as the set of nodes whose capacity exceeds $N^{\epsilon(N)}$. This formal definition makes of course sense only in the asymptotical behavior of the graph as $N \to \infty$. Intuitively, the core consists of 'big' nodes whose capacity 'is a power of N'. The node v^* with largest capacity satisfies $\Lambda_{v^*} \approx N^{1/\alpha}$.

Most capacity is at the bottom: Let $\gamma \in (\epsilon(N), 1/\alpha)$. The aggregated capacity of all nodes i satisfying $\Lambda_i > N^\gamma$ is proportional to $N^{1-(\alpha-1)\gamma}$ for large N. Thus, most of the capacity of any top section of the core resides near its lower boundary.

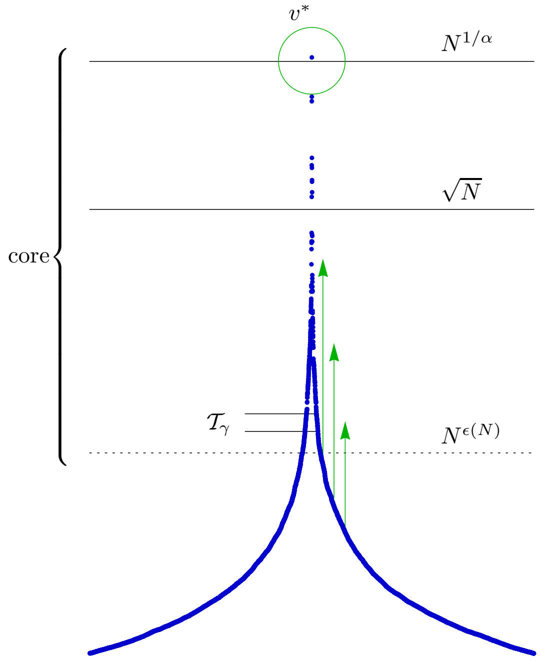

Fig. 1. The soft hierarchy structure of the core of an IVPLRG. Capacities are drawn from distribution (1) and plotted on logarithmic scale, highest in the center. The green arrows indicate how mighty neighbors a node typically has. Nodes bigger than $\sqrt{N}$ form almost a clique. Horizontally cut 'tiers' are denoted by $\mathcal{T}_\gamma$.

Density increases from low up to a clique: Consider thin horizontal sections, 'tiers' of the core, denoting

$$\mathcal{T}_\gamma = \left\{ i \in V : \ \Lambda_i \in (N^\gamma, N^{\gamma+\epsilon(N)}) \right\}.$$

When γ increases from $\epsilon(N)$ upwards, the internal edge density of $\mathcal{T}_\gamma$ grows from fairly low up to 1, which is obtained when γ passes the value $\frac{1}{2}$ and corresponds to a clique (a totally connected subgraph). Even the lower tiers of the core are almost connected internally.

The hierarchy and its height: Consider a core node v with capacity N^γ. The largest capacity found among v's neighbors is close to $\min(N^{\gamma/(\alpha-1)}, \Lambda_{v^*})$. Thus, the core network forms a 'soft' hierarchy in the sense that almost every core node (except v^*) has a neighbor with this much higher capacity. Iterating this observation, one obtains that the distance from v to v^* is with high probability at most

$$k^* := \left\lceil \frac{\log \log N}{-\log(\alpha - 1)} \right\rceil.$$

Thus, any two nodes of the core are with high probability connected by a path containing at most $2k^*$ hops and having an up-down profile in terms of capacities of the intermediate nodes.

The core is found quickly: Although the size of the core and its aggregated capacity are both asymptotically negligible parts of the whole, the latter number is sufficiently high to make the core attractive enough that a randomly chosen node is, with high probability, either connected to the core with still less than k^* hops, or not connected to it at all. Both alternatives have positive probabilities. The connected component containing the core is called the giant component.

The core is robust: If all nodes with capacity higher than N^γ are deleted, with a fixed $\gamma > 0$, the relative size of the giant component remains asymptotically unchanged, and the shortest paths are increased by a constant that does not depend on N.

Layers: We define layers of nodes, based on capacities $\mathbf{\Lambda}$:

$$\begin{aligned}
V_1 &= \{i \in V : \Lambda_i \geq N^{\beta_1(N)}\} - \{i^*\} \\
V_2 &= \{i \in V : N^{\beta_1(N)} > \Lambda_i \geq N^{\beta_2(N)}\} \\
&\cdots \\
V_k &= \{i \in V : N^{\beta_{k-1}(N)} > \Lambda_i \geq N^{\beta_k(N)}\} \\
&\cdots
\end{aligned}$$

where we used the notations: $V = \{1, 2, \cdots, N\}$ and

$$\beta_0(N) = \frac{1}{\alpha} + \frac{\epsilon(N)}{\alpha - 1}$$
$$\beta_j(N) = (\alpha - 1)\beta_{j-1}(N) + \epsilon(N),\ j = 1, 2, \cdots$$

The soft-hierarchy indicates that, for a node in layer i, there exists a neighbor in layer $i-1$, with probability tending to 1 as $N \to \infty$. Thus paths from the bottom of the layer system, layer number k^*, to the top node are no longer than k^*. This is probably also a tight bound, see [3].

III. A SCHEME FOR THE POWER-LAW GRAPH AS A DISTRIBUTED HASH TABLE (P-DHT)

A distributed hash table is a network for content search and storage, based on a hash function that turns user-friendly names to long binary strings. This hash function is common information, so any member of the network can produce a hash value corresponding to any name. The usual approach, like Chord [7], assumes a structured network. A power-law graph is not structured, since there is no overall picture of its topology. However, the core can be in some extent considered as a structure that emerges with high probability as the system grows, although an entirely random structure is not plausible for practical use and is replaced by another scheme later on. However, the theoretical analysis is simpler for a random structure and that is why we use it.

We use the core as a basis for the storage and retrieval of a content. We assume a static setting with large enough number of nodes, the dynamical case will be discussed later. For scaling, we assume that each node has one file and a corresponding hash value to be stored. Then in ideal case each node would have one hash entry to be stored. Let us define a hash entry as the hash value of a content and the corresponding network address, where the content resides. The hash values are usually modeled as Bernoulli type sequences of 0 and 1, where each value 0 or 1 is taken with equal probability and independently of other values. For the sake of tractability, we assume that the hash values fill the hash ring perfectly uniformly. Of course, in real systems this should be replaced with a Bernoulli type distribution, with some fluctuations of hash value density in the ring. This would complicate our analysis, probably without changing the principal indications.

In our scheme the hash values are stored in a tree-like subgraph of the core of the Poissonian power-law random graph. The principle for the tree construction is as follows. The root of the tree is the largest capacity node i^*. It is unique with probability 1. The next level of the tree, from the root, is formed by those layer 1 nodes that have a link to i^*. The third level is formed by some layer 2 nodes. Selection of those nodes is done according to the following rule. All nodes from layer 2 that have at least one link to a layer 1 node mark uniformly one of such link of theirs. If the marked link leads to the node that belongs to the tree so far (i.e. has link to i^*), then such a layer 2 node with its marked link is added to the tree. Then the procedure is continued layer by layer similarly to one in level 3. At each stage the procedure constitutes in analogous selection of new nodes and links to the tree constructed so far. As a result, the links in the tree are always between subsequent layers. This tree is constructed until it reaches the layer k^*. It may also happen that the tree terminates before it reaches the level k^*, we let this happen because otherwise the analysis becomes much more complicated.

Now we describe the principle of mapping the hash ring to the tree, see Fig. 1. The node i^* has some tree neighbors in layer 1, whose random number is denoted as Δ_1. This root node simply divides the whole circle of hash values in equal intervals between all its Δ_1 tree neighbors. This allocation of neighbors on the ring can be done internally, only node i^* knows it. A layer 1 node k within the tree has some random number of tree neighbors, $\Delta_2(k)$, within layer 2. Such nodes divide the hash ring again in even intervals between their neighbors, and so forth until the last layer k^*, where the corresponding tree degree is denoted by $\Delta_{k^*}(i)$, where i is a node in layer $k^* - 1$. This regular setting in ring allocation is made for sake of tractability. In a real algorithm a more robust way is needed, say, uniform distribution on the ring.

A storage of a hash entry starts from the root, and is based on the hash value. The entry is stored at some leaf node of the tree subgraph. This leaf node is uniquely defined by the hash value. Each node has a functionality of distributing the original hash values evenly on its hash table, a permutation of the original hashes. Each node then applies its own permutation to the original hash, and the hash entry is forwarded to a tree neighbor that has the closest internal hash value to the permuted one in counter clock direction of the node's internal hash ring, described in the previous paragraph. The original hash values can not be used for this purpose, since the forwarding rule makes them localized in a certain range. However, the new hash value must be a unique function of the original one. It is also necessary that nodes have independent ways of creating new hash values. The even distribution of permuted hashes is also an idealization. More practically some function, say, iteration of the hash function could be used. The original hash entry is thus stored at the leave node, which is at most in the layer k^*. From the construction it is obvious that each hash value is stored in a unique leave node.

The search is performed in an analogous way as storing. The request message travels along the same path as the storage request for the corresponding hash entry used. As the request reaches the target node, the information about the hash value is sent to the node that initiated the request. In an IP network, this information, stored in the leaf node, is simply the IP-address of the node that has the content or resource corresponding to the hash value. In other cases the information could be the shortest path from the source to the top of the hierarchy. Since the largest node is found with k^* steps, simply by following links upwards in hierarchy, the search paths have length $\propto 2k^*$.

IV. Table sizes of P-DHT

We use the above described conditionally Poissonian power law graph model to estimate the size of a typical hash table that a node has in the scheme of P-DHT described in previous section. We first made all estimates conditionally to drawn sequence of capacities, noted as $\mathbf{\Lambda} = \Lambda_1, \Lambda_2, \cdots, \Lambda_N$.

Main result and estimates conditional to capacities

We first consider conditional to $\mathbf{\Lambda}$ random graph and construct the P-DHT scheme on it. It should be noted that despite the capacities are under conditioning, the links are drawn, by

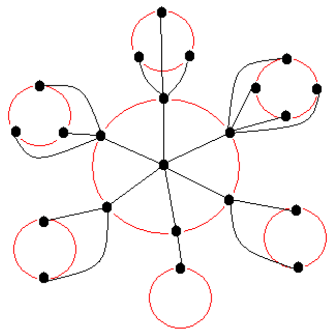

Fig. 2. A scematic picture of our DHT. Black dots represent nodes, links the tree neighbourhood. The top node i^* is at the center. The red circles represent the hash rings. First the top node places its neighbors on the hash ring, this is the red ring around the top node. Each of these neighbors replaces their down-hill neighbors on corresponding hash ring, depicted by the subsequent red rings. This continues until the leaf nodes are reached.

definition, independently of each other. To keep things simpler we particularize the definition of the tree. As we said a node in layer k selects one link to layer $k-1$, in the case it has at least one such link. Otherwise it remains outside the tree. It is simpler to make these links slightly more thin by reducing the capacities of nodes to the level of the bottom of the corresponding layer. In other words, in this three construction it is assumed that layer k nodes have capacity equal to $N^{\beta_k(N)}$. Formally this can be done by cutting the existing links, independently of each other, with some probability in the following way. The nodes with $\Lambda_\kappa, \kappa \in V_k$ and with $\Lambda_{\kappa'}, \kappa' \in V_{k-1}$ have link with probability $p_k(\kappa, \kappa') = 1 - \exp(-\Lambda_{\kappa'}\Lambda_\kappa/L_N)$. Define $p_k = 1 - \exp(-N^{\beta_{k-1}(N)}N^{\beta_k(N)}/L_N)$, $p_0(\kappa, \kappa'; k) = p_k/p_k(\kappa, \kappa')$ and obviously $0 \le p_0 \le 1$ for all such pairs of nodes κ, κ'. Considering subsequent layers V_k and V_{k-1}, mark pairs of nodes $\{\kappa, \kappa'\}$, where $\kappa \in V_k$ and $\kappa' \in V_{k-1}$, with the corresponding probability $p_0(\kappa, \kappa'; k)$, independently of each others. A link is called marked, if the corresponding pair was marked. Then our tree construction is simply such that the links in trees are taken uniformly out of marked links. As a result, the probability that a marked link exists between such nodes is just $p_{0,k} = 1 - \exp(-N^{\beta_{k-1}(N)}N^{\beta_k(N)}/L_N)$.

Finally, the probability that a node in layer V_k has at least one marked link to be chosen is $p_k = 1 - (1 - p_{0,k})^{v_{k-1}}$. It is

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7724
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7724

allowed that leaves of the tree are higher in the hierarchy than at the last layer. This could happen if a node in some layer $k < k^*$ is attached to a tree, but no node in layer $k-1$ selects this node as its tree neighbor. To analyze such events we would need a complete probability distribution (multinomial), which we postpone to future work. Here we just look at the typical case, and not at the worst case.

We want to find out the magnitude of the share of the hash entries, ρ_{k^*}, that come to a leaf node in which we enter by selecting successive uniformly random neighbors at the tree.

Consider the following sequence of random variables, and sequence of random nodes. Let Δ_1 denote the number of tree neighbors of the largest capacity node i^*. These are, if any, in the V_1. Then take any such tree neighbor in V_1 (with uniform probability) and denote by Δ_2 the number of its tree neighbors. All nodes that contribute to Δ_2 are in V_2. And so forth, until we get to layer V_{k^*} with corresponding random number Δ_{k^*}. Denote by ρ_{k^*} the random variable $\rho_{k^*} = N/\prod_{i\le k^*}\Delta_i$. This is the share of the hash values a leaf node in k^* gets, because the root node divides the hashes in Δ_1 equal sets, and so forth. Denote by v_k the cardinality of the layer V_k. The main result is the following:

Theorem 4.1: $\mathbb{P}(\rho_{k^*} \le m(N)) \to 1$, where $m(N)/\log\log N \to 0$ and $\log\log\log N/m(N) \to 0$, as $N \to \infty$,
which we shall prove at the end of this section.

For the proof of Theorem 4.1 we need several lemmas, starting from the following:

Lemma 4.2: Conditionally on $\mathbf{\Lambda}$, and $\forall\xi$ such that $\frac{1}{2} < \xi < 1$:

$$\mathbb{P}\left(\rho_{k^*} \le \frac{N}{v_{k^*}}\frac{1}{\prod_{i\le k^*} p_i(1-y_i^{\xi-1})}\right)$$
$$\ge 1 - \sum_{i\le k^*} e^{-\frac{1}{2}(y_i)^{2\xi-1}},$$
$$y_1 = p_1 v_1,\ y_i = \frac{p_i v_i}{v_{i-1}},\ i>1$$
$$p_1 = 1-\exp(-N^{\beta_1}\Lambda_{i^*}/L_N)$$
$$p_i = 1-(1-p_{0,i})^{v_{i-1}},\ i>1$$
$$p_{0,i} = 1-\exp(-N^{\beta_i+\beta_{i-1}}/L_N),\ i>1.$$

Proof: It is clear that Δ_1 is binomially distributed: $\Delta_1 \sim Bin(v_1, p_1)$, because all nodes in V_1 are potential neighbors of i^* and thus can belong to the tree. The probability that they are neighbors of i^* is given by p_1, using thinned links between V_1 and i^*. Similarly, all other variables Δ_i, $i>1$ are binomially distributed: $\Delta_1 \sim Bin(v_i, p_i)$. Indeed, potential neighbors are from layer i, thus $n = v_i$ in notation $Bin(n,p)$. For a node in layer i, the probability that it has at least one marked link to layer $i-1$ is $1-(1-p_{0,i})^{v_{i-1}}$, if such marked links exists. Then a uniformly randomly target is selected from layer v_{i-1}, and the probability that a given node is selected is thus $1/v_{i-1}$. The product of these two probabilities gives the parameter p. We obviously have $\mathbb{E}\Delta_i = y_i$. That is why Chernoff's bound gives $\mathbb{P}\Big(\Delta_i \ge y_i(1-y_i^{\xi-1})\Big) \ge 1-e^{-\frac{1}{2}y_i^{2\xi-1}}$, see [8], for any ξ, $1/2 < \xi < 1$. Then we have

$$\mathbb{P}\left(\prod_{1\le i\le k^*}\Delta_i \ge \prod_{1\le i\le k^*} y_i(1-y_i^{\xi-1})\right)$$
$$\ge \mathbb{P}(\forall i,\ \Delta_i \ge y_i)$$
$$\ge \prod_{1\le i\le k^*}\mathbb{P}\Big(\Delta_i \ge y_i(1-y_i^{\xi-1})\Big)$$
$$\ge \prod_{1\le i\le k^*}(1-e^{-\frac{1}{2}y_i^{2\xi-1}})$$
$$\ge 1-\sum_{1\le i\le k^*} e^{-\frac{1}{2}y_i^{2\xi-1}}.$$

It remains to note that the argument of the left-hand side is

$$\begin{aligned}\prod_{1\le i\le k^*}\Delta_i &\ge \prod_{1\le i\le k^*} y_i(1-y_i^{\xi-1})\\ &= v_{i^*}\prod_{1\le i\le k^*} p_i(1-y_i^{\xi-1}).\end{aligned}$$

■

We also need the following lemma.

Lemma 4.3: $\prod_{1\le k\le k^*}(1-N^{-m\beta_k}) \ge c > 0$, as $N\to\infty$, where $m>0$.

Proof: Let $k' = \max\{k : N^{m\beta_k} \ge \log\log N\}$. For large enough N, β_k is a monotonously decreasing function of k. Therefore, $N^{m\beta_{k'+1}} < \log\log N$. Then $m\beta_k < \log\log\log N$, for large enough N. $\beta_{k'+1} = \frac{(\alpha-1)^{k'+1}}{\alpha} + c'\frac{l(N)}{\log N}$, where c' is of the order of one. It is clear that the term with $l(N) = \epsilon(N)\log N$ is small and can be neglected, because $l(N)/\log\log\log N \to 0$. That is why we have $k' \le \frac{1}{\log\frac{1}{\alpha-1}}(\log\log N - \log^4 N)$, where $\log^4 N$ denotes 4-times log, and

$$\prod_{1\le k\le k'}(1-N^{-m\beta_k}) \ge \left(1-\frac{1}{\log\log N}\right)^{k^*+1-c\log^4 N}$$

$$\to \alpha-1>0.$$

The rest of the product can be estimated as

$$
\begin{aligned}
&\prod_{k'+1\le k\le k^*}(1-N^{-m\beta_k}) \ge (1-N^{-m\beta_{k^*}})^{k^*+1-k'}\\
&\ge (1-N^{-fm\epsilon(N)})^{1+c\log^4 N} \to 1,
\end{aligned}
$$

because $\log^4 N/N^{\epsilon(N)} \to 0$, where $f, c > 0$ are some constants. As a result, the whole product has a strictly positive lower bound. ■

Estimates unconditional to capacities and the proof of the main result

So far everything was conditioned on $\mathbf{\Lambda}$, now we try to get unconditional results. The total capacity L_N is a random variable with finite mean. The Markov inequality gives us the following:

Lemma 4.4: For any $\xi > 0$,

$$
\mathbb{P}\left(N \le L_N \le \frac{\alpha}{\alpha-1}N^{1+\xi\epsilon N}\right) \ge 1 - N^{-\xi\epsilon(N)}.
$$

The random variable L_N causes some formal difficulties. Indeed, under the condition that $L_N \in A$, where A denotes the range of values, say, in Lemma 4.4, some other random variables can become dependent. This is true for variables like v_k etc. To cope with this we use the following obvious inequality:

Lemma 4.5: Let A and B be any measurable sets, with complements $\bar{A}$ and $\bar{B}$. Then

$$
\mathbb{P}(A \cap B) \ge 1 - \mathbb{P}(\bar{A}) - \mathbb{P}(\bar{B}).
$$

If we can show that the right hand side goes to 1 as $N \to \infty$, then so does the left hand side.

Lemma 4.6: $\exists$ some $f > 0$, such that:

$$
\begin{aligned}
&\mathbb{P}\left(\frac{1}{2}\mathbb{E}(v_k) \le v_k \le 2\mathbb{E}(v_k)\right)\\
&\ge 1 - 2e^{-\frac{1}{8}\mathbb{E}(v_k)} \ge 1 - 2e^{-N^f}.
\end{aligned}
$$

Proof: This follows from the fact that v_k is a binomially distributed random variable. Then the first inequality is the Chernoff's bound. It remains to calculate the expectation of v_k. We have $\mathbb{E}(v_k) = NN^{-\alpha\beta_k}(1 - N^{\alpha(\beta_{k-1}-\beta_k)}) \ge N^f$. It is easy to see that $1 - \alpha\beta_k > 0$, and $\alpha(\beta_{k-1} - \beta_k) \le 0$. Thus we can select some $f > 0$ so that the inequality holds. ■

Let us denote by C the event $\{\frac{1}{2}\mathbb{E}(v_k) \le v_k \le 2\mathbb{E}(v_k)\}_{1\le k\le k^*}$. Then, Lemma 4.5 gives us the needed estimate for the joint probability and we get

Corollary 4.7: $\mathbb{P}(C \cap L_N \in A) \ge 1 - 2k^*e^{-N^f} - N^{-\xi\epsilon(N)}$.

The conditional probabilities $p_i, i = 1, 2, \cdots, k^*$ are also random variables. We have the following:

Lemma 4.8: There exist some constants $s, r, c, f > 0$, s.t.

$$
\begin{aligned}
&k = 1, 2, \cdots, k^*,\\
&\mathbb{P}\left(\{p_k \ge 1 - e^{-sN^{r\epsilon}}\}_{1\le k\le k^*}\right)\\
&\ge 1 - 2k^*e^{-cN^f} - N^{-\xi\epsilon(N)}.
\end{aligned}
$$

Proof:

By definition, $p_k = 1 - e^{-v_{k-1}N^{\frac{\beta_k+\beta_{k-1}}{L_N}}}$. Let us denote by B the event that $\{p_k \ge 1 - e^{-sN^{r\epsilon}}\}_{1\le k\le k^*}$. Then we have: $\mathbb{P}(B) \ge \mathbb{P}(B \cap L_N \in A \cap C) = \mathbb{P}(L_N \in A \cap C) \ge 1 - 2k^*e^{-N^f} - N^{-\xi\epsilon(N)}$, where the last inequality comes from Corollary 4.7. The last equality comes from the fact that B is true when both C and $L_N \in A$ take place. Indeed, under these conditions

$$
\begin{aligned}
&p_k \ge 1 - e^{-2\frac{N^{1-\alpha\beta_{k-1}+\beta_k+\beta_{k-1}}}{2N^{\frac{(\alpha-1)}{\alpha}}}}\\
&\ge 1 - e^{-gN^{\beta_k-(\alpha-1)\beta_{k-1}}} \ge 1 - e^{-gN^{(\alpha-1)\epsilon(N)}}
\end{aligned}
$$

for some $g > 0$.

■

As a result we conclude that the estimates given in Corollary 4.7 and Lemma 4.6 are true with probability tending to 1, or a.a.s. Indeed, it is clear that $k^*e^{-N^f} \to \log\log Ne^{-N^f} \to 0$.

Now we can prove Theorem 4.1:

Proof: Let M denote the event that $\rho_{k^*} \le \frac{N}{v_{k^*}}\frac{1}{\prod_{i\le k^*}p_i(1-(\frac{p_iv_i}{v_{i-1}})^{\xi-1}}$, when we make a uniformly random selection of ρ_{k^*}. Let M_0 be such a subset of M that $\rho_{k*} \le m(N)$, with function m as in the conditions of the theorem. We have:

$$
\begin{aligned}
&\mathbb{P}(M_0) \ge \mathbb{P}(M_0 \cap C \cap L_N \in A)\\
&= \mathbb{P}(M_0 \mid C \cap L_N \in A)\,\mathbb{P}(C \cap L_N \in A)\\
&\to \mathbb{P}(C \cap L_N \in A) \to 1,
\end{aligned}
$$

because $M \cap C \cap L_N \in A = M_0 \cap C \cap L_N \in A$. Indeed, this can be easily verified by substituting the conditions that event $C \cap L_N \in A$ takes place to the right hand side of the condition of $M \supset M_0$, and then using Lemma 4.3 for the product. That is why we can use the Lemma 4.2 for $\mathbb{P}(M \mid C \cap L_N \in A)$, showing that it goes to 1 as $N \to \infty$, and Corollary 4.7 to show that also $\mathbb{P}(C \cap L_N \in A)$ has the same limit. ■

V. Model for graph process with power law degrees

A random graph is hardly a good choice for a network topology. That is why we consider how one could get a topology that has properties of a power law random graph, without

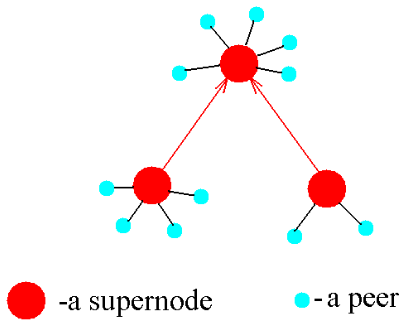

Fig. 3. As schematic picture of the topology of supernode tree with attached peers. Red arrows repreresent uplinks between supernodes.

some drawbacks, say, more than one connected components. Such topology should also be 'self-organizing', it is formed by local actions of peers, without global steering elements. Here we give a sketch of one possibility that seems to be well suited for our purpose. We postpone more careful analysis of this model.

Our construction relies on generalized Pólya's urn process by Chung et al [9] and on the concept of soft hierarchy created for random power law graph. We assume dynamical setting where nodes can come and go. First consider only node arrivals, the leaves require a certain modifications that are given in the next sections. An arriving node makes a decision whether it will be a supernode or not. With a fixed probability p it takes the role of the supernode and with probability $1-p$ it will be a peer. Each peer in the system is attached by a link to one supernode. The arriving node that is going to be a peer, samples randomly and uniformly over all peers in the system and attaches to the same supernode as the sampled peer. If the arriving node decides to be a supernode it does the same random sampling and makes the link to the corresponding supernode, in addition it is considered as a supernode with one attached peer, that is itself. The last circumstance is needed to make the newly born supernode visible for future arriving peers.

It appears that this arrival process is just another formulation of Chung's et all, urn process. Indeed, this discrete time urn process assumes that at each time step with probability p a new bin is created and one new ball is put into it, and with probability $1-p$ a ball is sampled uniformly among all existing balls and one new ball is placed in the same urn as the sampled one. Obviously creation of a supernode is equivalent to creation of a bin and so on. It was shown that in the limit of large time, the fraction of bins with $i \in \{1,2,\cdots\}$ balls, f_i converges to

$$f_i \sim i^{-(\alpha+1)}, \alpha = \frac{1}{1-p}.$$

Thus, in the range $0 < p < \frac{1}{2}$, we get the power law distribution with finite mean and infinite variance. As a result, the distribution of number of peers attached to a supernode follows this same law in the limit of large system. To achieve soft hierarchy of supernodes the following procedure must be added. When a supernode gets a new neighbor, it makes new uniform sampling of peers in the system, if the found peer is attached to a supernode with higher degree than the one to which the initiator of the sampling was currently attached, then it swaps its supernode contact, in other wards cuts the old link and attaches to a higher degree supernode. In the case that a supernode with lower or equal degree was found, nothing happens. This results in a supernode subgraph, that is obviously a tree. We argue that it has low typical height of the order of $\log\log N$, where N is number of supernodes. The only operation that is needed to create it is the uniform sampling of the peers, this can be done in a distributed manner and as a result the supernode hierarchy is self organized.

In order to make our model tractable, we simplify it. It is well known that the expected degree sequence of a power law graph has the form:

$$d_j = (t/j)^{\frac{1}{\alpha}},$$

where, t is the order of graph, $j = 1,2,\cdots t$. In our case, t is the number of arrivals so far. We conjecture, that in the limit of large t, it is plausible to assume that the degree sequence follows the above relationship, and results concerning distances in the network are asymptotically identical. That is why we assume deterministic degrees or 'capacities', that grow in time, it is assumed that a supernode gets a new neighbor when its capacity reaches the level of the subsequent integer. Thus we should use floor function for the degrees in above relation. As a result degree of supernode with number k at time $t > k$ is

$$d_j(t) = c\left\lfloor\left(\frac{t}{j}\right)^{\frac{1}{\alpha}}\right\rfloor,$$

where c is a constant. We agree, that the degree of a supernode does not include links between supernodes. However, a supernode has at least one peer attached to it, indeed in the moment of creation a supernode has one peer attached to it. The sum of all supernode degrees equals to t.

In this deterministic degree models, we can estimate probability of supernode-supernode links. As in original model, when a supernode is created it samples all peers in the system uniformly and makes link to that supernode to which the sampled peer was attached. This link we call the 'uplink'. Each time the supernode's degree increases by 1, the supernode repeats this procedure and updates the uplink only if it founds a supernode that has larger degree than the current uplink target has. Denote by $\mathbb{P}(j \to k, t)$, the probability, that supernodenode j has uplink to supernode k at time t. We have the following suggestion:

Hypothesis 5.1:

$$\mathbb{P}(j \to k, t) \geq 1 - \exp(-md_j(t)d_k(t)/t), j > k,$$

where $m > 0$ is a constant. We give only a sketch of proof. Node j makes $c(t/j)^{1/(\alpha)}$ attempts to improve its uplink target. Denote moments of time when these attempts happen by $s(i)$, $i = 1, 2, \cdots, c(t/j)^{1/(\alpha)}$, where we neglect the notion of floor function, which should be plausible in large time limit. All these times are upper bounded by t. Thus we have, with some positive constant m': $1-\mathbb{P}(j \to k, t) = \prod_{i=1}^{c(t/j)^{1/(\alpha}} (1-m'(s(i)/k)^{1/(\alpha)} \frac{1}{s(i)}) \leq (1-\frac{m'}{k^{1/(\alpha)}} \frac{1}{t^{1-1/(\alpha)}})^{(t/j)^{1/(\alpha)}} \leq \exp(-m'(t/j)^{1/(\alpha)}(t/k)^{1/(\alpha)}/t)$, from which the claim follows.

In previous chapters a power law random graph was studied. In such models link probability was given by an analogous formula and with replacing the magnitudes like $d_j(t)$, with power law distributed random capacities. In our current models we keep only one uplink, but make many trials to improve them. Roughly speaking, if the all such trials would result in a link between supernodes, we would have similar graphs in the large t limit. Then the mentioned soft hierarchy with low height was found as an asymptotic structure. It was also recognized that within the core of large degree nodes, it is sufficient to look just one link that goes from a given vertex to the largest capacity neighbor. Our hypothesis is that the above uplinks are similar to those 'extreme' links.

Another feature in the original random graph was that the capacity of a set of nodes can be added up, in the sense that the probability that a node has link to such set of nodes is given by an analogous formula with capacity of the set of nodes represented with the sum of capacities over that set. We have a similar result here. Let denote by $\mathbb{P}(j \to S, t)$ that a node in has an uplink to a node in a set of nodes S, with $s \in S \Rightarrow s < j$. Then we have

Hypothesis 5.2:

$$\mathbb{P}(j \to S, t) \geq 1 - \exp\left(-md_j(t) \sum_{s \in S} d_s(t)/t\right).$$

Proof: This follows from the Hypothesis 5.1, since $1 - \mathbb{P}(j \to S, t) = \prod_{s \in S} (1 - \mathbb{P}(j \to s, t)) \leq \prod_{s \in S} \exp(-md_j(t)d_s(t)/t) =$ $\exp(-md_j(t) \sum_{s \in S} d_s(t)/t)$, which was the claimed inequality. ∎

Now we can repeat definition of approximately $\log \log t$ layers of supernodes, such that in the limit of large t, a node in layer k has asymptotically almost surely an uplink to the next upper laying layer $k - 1$. Thus, in such a way one can show the $\log \log t$ distance scaling to be hold. Of course, more rigorous arguments are needed, however it seems that the result is very likely to be right. The distance distribution show a one peak function, the position of this peak, the typical distance, is almost a constant corresponding to $\log \log t$-type dependency. The tail of the distribution grows somehow quicker, corresponding to $\log t$ scaling of the diameter, some rare pairs of nodes have such record distances.

We postpone the corresponding rigorous analysis. Although such analysis would back up our model theoretically it has a limitation that the results are only asymptotical in t. Now more important is what kind of graphs we get in a reasonable range of t, say up to millions of peers in the system. That is why the simulations have also important role in this perspective, although they have very little to give for rigorous analysis.

VI. Soft hierarchy tree instances

We made some computer experiments with the proposed algorithm. They suggest that the results are in qualitative agreement with the connections to the power law random graph model suggested in previous sections. The distance scaling is very good indeed, the distances hardly grow in the range of thousands and millions of peers. This is even notable if compared with the $\log t$-scaling typical for networks based on distributed hash tables, like the Chord, where distances are scaled somehow similarly to the hypercube with dimension t. In very large networks the path lengths become rather long, say, of the order of 20, which can be untolerable. Our algorithm produces graphs, in any reasonable range, with typical hop count distances less than 10 with parameter $p = 0.1$. Only few node pairs have distances that correspond to the diameter, which has asymptotical scaling of $\log t$, [2]. In this respect already the supernode graph shown in Fig. VI is somehow representative. Indeed, the typical distances are only a few hops and most of the supernodes have only very low number of supernode neighbors while there are a few very high degree nodes in the system. Such heavy nodes are crucial for the connectivity and have obviously a central role in mediating between supernodes. In a real system it would be necessary to have in places of such heavy supernodes some sufficiently capable and reliable entities given by the provider. On the other hand, typical supernodes are lightly involved in the system and could well be handled by voluntaries. This picture will remain unchanged even if the size of the system grows with many orders of magnitude, remainig almost static. Again a typical peer has to deal with a light burden, while the heavy nodes have the main burden in increasing connectivity. However, even for these large nodes the needed degrees are sub linear functions of t, thus making our approach clearly different from the centralized server based solutions. Indeed, we have the following suggestion:

Hypothesis 6.1: In a power law supernode tree, the degree of a large enough supernode of rank i at time t scales at most as $d_j(t)^{2-\alpha} = (t/i)^{(2-\alpha)/\alpha}$.

Obviously, in our case the power of t fulfills $0 < (2-\alpha)/\alpha < 1$. This result follows from an estimate for expected degree of such a high rank node, and the law of large numbers telling that the deviations from this value can not be too large with notable probability. The reasoning goes like this

$$\begin{aligned}&\mathbb{E}\,|\{j : j \to i, t\}|\\ &\geq d_i(t) \sum_{j>i} d_j(t)/t = (t/i)^{1/\alpha} t^{1/\alpha} \sum_{j>i} j^{-1/\alpha}/t\\ &\approx (t/i)^{(2-\alpha)/\alpha}.\end{aligned}$$

The following computer experiments (see Fig. 5) support the view of good distance scaling in the power law tree. We take values of $t = 10^4, 10^5, 10^6$, the mean distance remaining almost constant around 5. The diameter shows a bit larger values $(13-18)$, but the numbers of node pairs within such large distance are insignificant.

VII. Coping with 'churn'

So far we assumed a monotonous growth of the network. This is motivated by the assumption that the presented system shows good scalability with large orders and is aimed for large scale systems, where the main challenge lays. However, the algorithm should also cope with decreasing number of nodes without centralized control. We argue that this is in principle possible. This part of the work is only a preliminary sketch. In future work we shall study this question in more details.

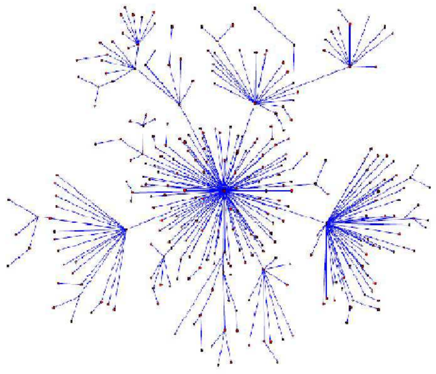
Fig. 4. An example of soft hierarchy tree with $p = \frac{1}{10}$, corresponding to $\alpha = 1\frac{1}{9}$, only the supernodes, 301 in numbers, are shown.

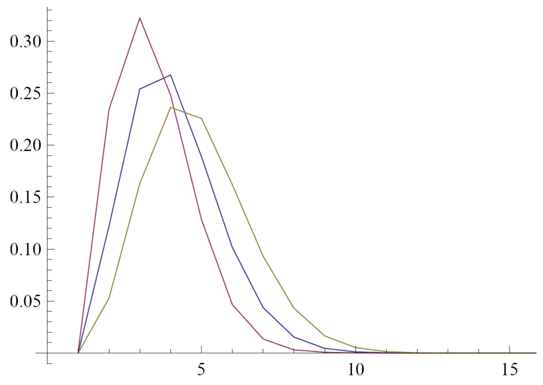

Fig. 5. Empirical hop count distance distribution in a soft hierarchy tree with $p = 0.1$ and $t = 10^4$ (red line), $t = 10^5$ (blue line) and $t = 10^6$ (green line). It shows fraction of pairs of nodes at distances $1, 2, \cdots$, found in computer experiment with 10 instances of two smallest graph sizes and 1 instance for the largest.

When the number of peers decreases in the system, this happens in homogeneous manner in the sense that all supernodes lose their peers in a uniform fashion. This is a result of unstructured design following from the uniform sampling of the peers. Thus a super-node can proceed autonomously, based only on its experience of its neighboring peer dynamics. That is why we suggest the following procedure of backtracking the uplinks. The supernode keeps track on its history of trials to change the uplink. It remembers the addresses of previous uplinks and the number of attempts that were needed to achieve the consequent improvement. For each departure of its neighboring peer, the supernode goes back in history, a new arrival on the other hand moves it forward to history, and when the balance of arrivals and departures is $+1$, it makes a new attempt to improve the uplink, setting all counters to 0. On the other hand, when the departures exceed the arrivals in a way that the supernode's counter hits the moment when the uplink was updated last time, it tries to change its uplink to this previous target.

Another type of failure occurs when a supernode leaves the network. Then we assume that the supernode that has this gone node as an uplink target changes the uplink to the previously used one, and if there is none, it makes a new trial to find an uplink target. However, we assume that the large nodes might be more robust than the typical nodes. Say, they are backed up by a provider. Then such failures of supernode may be reduced, and a typical supernode departure is one where a very light weight supernode leaves and does not require any massive procedures. Probably some other relaxations can be done, since the main target is to keep the soft hierarchy in up-to-date state. This means that nodes belonging to a certain layer has an uplink to the next layer. This circumstance is not violated immediately after some minor changes in the order of the system.

VIII. Finding a uniformly random peer

So far we did not specify how the process called 'find a peer uniformly over all peers in the system' is realized. Let us call this function FindUP. Again we provide only some sketches of solution. One could assume an auxiliary network for this purpose. For instance, in Chord type DHT this could be done, [10]. However, it would be nicer to have only one network.

We have obviously one natural 'randezvous point' in the network, the top node. It can be found in a distributed fashion by going along uplinks as described in previous sections. The requests for a random peer are forwarded to the top node. It could then, say, generate a sequence of random hashes. These hashes are used to forward the request the request to the random leaf node of the tree, at each step the subsequent random hash value is used to forward the request. The leaf node picks up uniformly, one of hash entries it possesses, an this entry is returned as the result of the request.

This FindUP function can also be used for peer joining process. A similar approach was used in [10]. It is necessary that an arriving peer gets an address of any supernode and forwards through it a FindUP request.

Acknowledgements This work was financially supported by the EU-project Net-Refound.

References

[1] Reittu, H., Norros, I.: On the effect of very large nodes in internet graphs. In: Proceedings of IEEE GLOBECOM'03, Taipei, Taiwan (2002)

[2] Chung, F., Lu, L.: The average distances in random graphs with given expected degrees. Internet Mathematics **1** (2003) 91–113

[3] van der Hofstad, R., Hooghiemstra, G., Znamenski, D.: Distances in random graphs with finite mean and infinite variance degrees. Electronic Journal of Probability **12** (2007) 703–766

[4] Norros, I., Reittu, H.: On a conditionally Poissonian graph process. Adv. Appl. Prob. **38** (2006) 59–75

[5] Norros, I., Reittu, H.: On the robustness of a power-law random graph in the finite mean, infinite variance region. Internet Mathematics, to appear (2009) Arxiv: 0801.1079.

[6] Norros, I., Reittu, H.: Network models with a 'soft hierarchy': a random graph construction with loglog scalability. IEEE Network **22(2)** (2008) 40–46

[7] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking **11**(1) (2003) 17–32

[8] Janson, S., Luczak, T., Ruciński, A.: Random Graphs. Wiley (2000)

[9] Chung, F., Handjani, S., Jungreis, D.: Generalizations of Polya's urn problem. Annals of Combinatorics **7** (2003) 141–153

[10] Norros, I., Pehkonen, V., Reittu, H., Binzenhfer, A., Tutschku, K.: Relying on randomness - PlanetLab experiments with distributed file-sharing protocols. In: Next Generation Internet Networks 3rd EuroNGI Conference, Trondheim, Norway (2007)