Fast Stochastic Analysis of P2P File Distribution Architectures

Damiano Carra
Dip. di Informatica e Telecomunicazioni
Università di Trento, Trento, Italy
carra@dit.unitn.it

Renato Lo Cigno
Dip. di Informatica e Telecomunicazioni
Università di Trento, Trento, Italy
locigno@dit.unitn.it

Ernst W. Biersack
Institut EURECOM
Sophia Antipolis, France
erbi@eurecom.fr

***Abstract*— In this paper we investigate which is the most efficient architecture and protocol that can be used for file distribution. The focus of the analysis is to understand not only the parameters that influence the distribution process (constraints on the number of neighbors, bandwidth heterogeneity, etc.), but also the impact of the peer behavior, such as selfishness or neighbor selection strategies. The analysis also compares different tree- and mesh-based distribution architectures. We developed an ad-hoc Monte-Carlo technique that is able to analyze scenarios with millions of peers, a network size that traditional discrete-event simulators are not able to treat. The results give an accurate view of the fundamental protocol parameters and policies that impact on the final performance and allow designers to devise improved protocols.**

I. Introduction

Peer-to-peer (P2P) networking has emerged as a powerful communication paradigm and it is gaining increasing attention. In this work we consider systems for *collaborative content distribution*, where the content is time-critical. Applications include, for instance, distribution of virus footprints or software updates. We assume a BitTorrent-like [1] distribution protocol where the content is divided in independent pieces called *chunks*. P2P systems are considered self-scaling. But self-scaling implies more than just adding resources as new users enter the system: it means finding proper algorithms and protocols to exploit them. The performance of distribution protocols and architectures is influenced by many parameters. A partial list includes: the number of chunks, the constraints on outdegree and indegree, the input probability density function (pdf) of the peer bandwidth (that characterizes the heterogeneity of the peers), the neighbor selection strategies, or the percentage of selfish peers. In a given scenario, what is the most efficient way to distribute a content to the users?

Our approach considers a high level characterization of the distribution process so that we can analyze it without focusing on the implementation details. We develop an efficient Monte-Carlo based solution technique that allows to compute the metrics of interest (e.g., the download time) for P2P systems with several million of peers. This approach is alternative to a detailed event-driven simulation of the corresponding P2P network. We deal only with building the distribution system given the general rules and properties of the peers. Our technique allows for detailed comparisons between different distribution architectures of very large size that were previously not feasible. Although they may look like fundamental results, to the best of our knowledge such a systematic study and comparison was never carried out for heterogeneous and realistic scenarios.

A. Related Work

Performance analysis in terms of the minimum time required to distribute a file using a P2P system has only in recent years received some attention. Analytical approaches proposed so far [2][3][4][5] (for a detailed analysis, please refer to our technical report [12]), as well as simulation based analyses [6], are focused on specific systems and not on a generic framework that allows the comparison of different distribution architectures. Moreover, only [2] and [5] tackle the problem of different access bandwidths among peers, which is instead treated in this paper, but the former does not consider architecture influence and the latter does not take into account mesh-based architectures.

A related topic where distribution architectures are explicitly taken into account is the delivery of streaming services through overlay multicast. ALMI [7] and SplitStream [8] define a set of mechanisms to efficiently distribute the streaming application to many overlay peers. They build in different ways distribution trees and manage the dynamics of leaving and joining peers. Nevertheless most of these studies are focused on protocol design and do not analyze the impact of the distribution architecture on performance.

In our previous works we have tackled and discussed specific aspects of the problem related to deterministic case [9] and heterogeneous case with chain-based architectures [10] using entirely different techniques.

II. Design Parameters and Distribution Architectures

A. Basic Model and Performance Metrics

Consider a scenario with $\mathcal{N}$ peers, where peers have different, symmetric or asymmetric, access link bandwidths. There is only one content source in the system with bandwidth at least equal to the highest peer bandwidth. All peers are independent, so we can consider the bandwidth of a peer i a random variable b_p with known density, which is identically distributed for all peers.

We focus on the distribution of a single file with a BitTorrent-like distribution protocol: the file is partitioned into C chunks. Each peer can start serving the file to another peer once it has completely received the first chunk. The file size is $\mathcal{F}$; the time needed to download the complete file with the lowest bandwidth in the network is referred to as $T_R = \frac{F}{\min(b_\mathrm{p})}$ and it is also called *one round*. We define the *eligibility time* t_i^{el} of peer i the time at which peer i can start to upload to other peers. t_i^{el} are random variables, since they depend on the peer bandwidths.

The signaling messages necessary to manage the dynamics of the overlay structure (join, leave, synchronization with neighbors, message used to build the distribution architecture) are negligible with respect to the file size, and no errors, failures or other bottlenecks other than the peer bandwidth are present.

Each peer i has a constraint on the maximum and minimum number of active uploads (the outdegree of the peer): $k_i^{\max}$ and $k_i^{\min}$. We define *step distance* or *step depth* of peer i, $d^{(i)}$, the number of hops from the root (content source) to peer i.

The main performance metrics are the *download time* T of the content, either for a given user i (T_i), or for the whole community (T_t), or the mean $\overline{T}$ of all the individual download times T_i.

B. Unbalanced and Uneven Trees

When distributing a content using a tree-based architecture, the resulting tree is, in general, a structure where the leaf peers do not all have the same distance (in terms of number of hops) from the root. The speed of growth of the different branches is not the same and the deeper branches are those that contain faster peers, i.e., peers with smaller eligibility times t^{el}. We call such trees "*uneven*."

The literature on tree-based distribution architectures normally considers trees where leaves have the same distance from the root. We call such trees "*unbalanced*." The difference between unbalanced and uneven trees is substantial: in an unbalanced tree, a slow peer will influence the reception of all peers in its subtree, in an uneven tree, a slow peer may not even have the possibility to have children. Since we are interested in the download time, it is worth to look at a weighted graph where the weight associated to a directed edge is given by the difference between the download times of the peers connected by the edge. Considering unbalanced trees, this representation shows the disparity in terms of download time among leaf peers that are at the same step distance. In Fig. 1 the weight is represented as a difference in edge length. Conversely, in uneven trees, leaf peers are at different step distances and the weighted graph gives a pictorial illustration why the tree grows in this way: a new edge is added only after a peer becomes eligible and this forces a uniform growth of the *weighted* graph.

C. General Mesh Architecture

Tree based architectures have known shortcomings. Each peer has only one ancestor and in case of a failure, the entire subtree will stop receiving data. Each peer must divide the upload bandwidth among its children, so children use only a fraction of their download bandwidths for receiving chunks; if we consider the case of asymmetric capacities, where the upload bandwidth is smaller than the download bandwidth (as in the case of ADSL), the percentage of unused download bandwidth increases even further. Finally, there are peers that have received the entire file without uploading a single chunk, resulting in unfairness and poor performance.

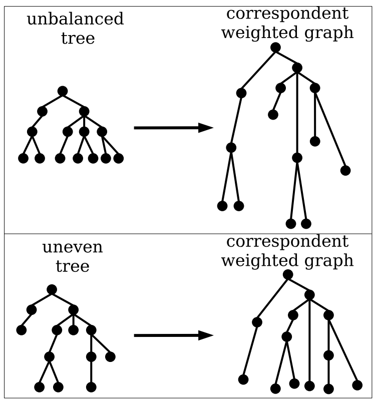

Fig. 1. Difference between unbalanced and uneven trees, considering the corresponding weighted graphs where edge length represents the download time.

Mesh based architectures are meant to overcome these problems. Peers can upload to other peers already reached by the content. In this case we have to consider the 'freshness' of the information that a peer is downloading from its fathers. We assume that the server, that is the only node that has the full content, is able to differentiate what it is distributing, e.g., it gives the chunks in different orders to its children. We define *diffusion trees* the trees generated by these children (we call them *first generation children*): diffusion trees can overlap, i.e., peers can receive the content from different fathers provided that these fathers belong to different diffusion trees. In general, if the server has k_s first generation children, each node can have up to k_s fathers and each father can provide up to $\mathcal{F}/k_s$ fresh content. For instance, in Fig. 2, we have $k_s = 2$ first generation children and $C = 6$ chunks; each nodes can receive up to $C/k_s = 3$ fresh chunks from different fathers. For efficient distribution it is required that only leafs of diffusion trees, which are those peers that do not find any unreached peer among their neighbors, start behaving as "additional fathers." Note that a peer can receive from less than k_s fathers, since each father has the whole file (for instance, the node at the extreme left). For a detailed characterization of mesh based architectures, where different cases are analyzed, refer to [12].

III. Numerical Solver

We first describe the generic solution for mesh architectures. In case of tree-based architectures we use a faster implementation that takes into account the particular structure of the problem. We then discuss the complexity of both approaches.

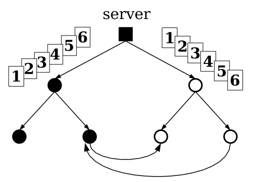

Fig. 2. Example of possible chunk orders when distributing the content ($C = 6$, $k_s = 2$).

A. Basic Behavior Description

The basic algorithm first builds the diffusion trees and then analyzes the mesh approach starting from the leaf peers of each diffusion tree. It is possible to block the subtree overlapping, obtaining the performance of tree based architectures. The following paragraphs give a detailed description of the basic behavior.

Input parameters. The main input is the pdf of the peer bandwidths. Other input parameters of interest are the number of peers $\mathcal{N}$, the number of neighbors for each peer and the policy for neighbor assignment, the outdegree constraints k_i^{max} and k_i^{min}, and the number of chunks. Finally, it is possible to specify the maximum step distance allowed for the architecture. Imposing a strict bound on the distance, we can obtain unbalanced trees.

Initialization. The tool assigns the highest bandwidth to the root and to the first generation children. If the chunks of the file $\mathcal{F}$ are distributed in k_s different orders to obtain a mesh, then the root selects exactly k_s children to fairly compare results with different k_s. For each child, the tool computes the eligibility time and assigns the download rate.

Diffusion. Each peer i at level 1 (level 0 is the server) randomly selects peers in its neighborhood to upload to until its upload bandwidth is saturated, i.e., the sum of the download bandwidths of its children is greater than its upload bandwidth, or no peer without chunks are left, provided that the constraints k_i^{max} and k_i^{min} are met. Once the list of children is created, the ancestor calculates for each child i the eligibility time t_i^{el} and the rate r_i (the dimension of a chunk divided the time necessary to download it) according to the max-min fairness criterion. From the eligibility time (i.e., the time a peer finishes to download the first chunk) and the rate, we can compute the total download time of each child i: $t_i^{\text{download}} = t_i^{\text{el}} + \frac{\mathcal{F}}{C}(C-1)r_i$. If the peer has no children, it is placed in a list for next rounds analysis.

Cross connections. Leaf peers in the diffusion subtree are those that find no unreached peers among their neighbors. Leaves start to help their neighbors, provided that they belong to different diffusion subtrees. The process of neighbor selection is done as in the previous case, but here the *spare* upload bandwidth of the ancestor and the *spare* download bandwidths of the neighbors are considered. For each neighbor, knowing the eligibility time and the additional rate r_i^{add}, we can calculate the new, reduced download time: $t_i^{\text{download}} = t_i^{\text{el}} + \frac{\mathcal{F}}{C}(C-1)(r_i + r_i^{\text{add}})$. We suppose C sufficiently high so that the difference among the starts of different contributions is not significant. Additional cross connections are realized respecting the usual constraints.

End of the realization. The realization stops when no more cross connection can be done. In this state all the download times can be computed.

Stop criterion. The performance indices at the end of each realization are samples of known i.i.d. random variables, so that standard techniques can be used to estimate the confidence intervals of the whole histograms (see for instance [11]). The tool stops when all bins of the histograms have a $\pm 10\%$ relative confidence interval with a 0.95 confidence level.

B. Optimization for Tree-Based Architectures

For these architectures we can consider, instead of the complete tree, *portions*, or *sample paths* exploring the tree, and we infer results for the entire structure. A sample path is a path from the root to a leaf peer that registers the number of children selected at each step, then chooses randomly a child and continues the process. The distance from the root is an input and can be expressed as number of hops (step distance) or maximum t_i^{el}: the former results in unbalanced trees, the latter in uneven trees. By inferring the number of peers from the analyzed path, we have a sample that can be used to reconstruct the entire tree. We take several of these sample paths and we consider the same statistics and the same stop criterion of the full version of the tool. This can be seen as a semi-analytical technique, since the sample paths are simulated, while the statistical properties of the whole tree are derived analytically from the sample.

In a network with $\mathcal{N}$ peers, the simulator builds only a path of $\log_{\overline{k}} \mathcal{N}$ levels, where $\overline{k}$ is the mean outdegree, and for each level $\overline{k}$ peers are extracted on average, so the total number of selected peers for each iteration is $\overline{k} \log_{\overline{k}} \mathcal{N}$.

With this fast implementation we can derive results for up to 10^8 peers. To the best of our knowledge numerical results on P2P networks and distribution networks in the literature, rarely extend above 10^3–10^4 peers, with some specific cases reaching 10^5.

C. Solution Complexity

The complexity of the basic algorithm is linear, i.e., it is $\mathrm{O}(\mathcal{N})$, since every node must be analyzed at least once for each iteration. Nevertheless, in order to find the statistics, the simulator iterated the main routine until the stop criterion is reached. As the number of peers increases, the number of iterations decreases. In fact, if we consider the download time, we notice that it strongly depends on the minimum rate encountered in the path from the root to the peer. As the number of level increases, the minimum bandwidth encountered tends to the lowest possible bandwidth (the probability increases geometrically at each level). The variability of the measured values (for instance, the mean download time) then decreases and the desired confidence is reached in less CPU time. This means that the total complexity is $\mathrm{O}(\alpha(\mathcal{N}) \cdot \mathcal{N})$, where $\alpha(\mathcal{N})$ is a monotonically decreasing function.

For instance in the numerical examples in Sect IV convergence happens in less than 1,000 realizations for 10^4 peers, less than 500 for 10^5 and less than 200 for 10^6. For 10^6 peers and mesh-based architectures this means 4-5 hours of CPU on a standard PC, 10 – 20 minutes for 10^5, while for a smaller number of peers the time becomes negligible.

In the case of fast implementation, with similar arguments it is easy to show that the complexity is $O(\beta(\mathcal{N}) \cdot \log \mathcal{N})$, where $\beta(\mathcal{N})$ is a monotonically decreasing function.

IV. Numerical Results

As numerical example we consider a density function for the peer bandwidth summarized in Table I.

TABLE I
BANDWIDTH DISTRIBUTION USED IN THE EXAMPLES

Bandwidth	% peers
56 kbit/s	13%
640 kbit/s	23%
1.2 Mbit/s	64%

We use a number of chunks C equal to 100, but a sensitivity analysis with different values of C indicates a qualitative behavior independent of C, as long as $C \gg 1$. In reporting results, we normalize the data such that $\frac{|\mathcal{F}|}{\min(b_i)} = 1$ 'round', where $|\mathcal{F}|$ is the content size in bits and $\min_i(b_i)$ is the minimum bandwidth of the input pdf in bits/s.

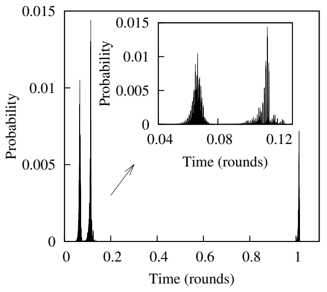

Fig. 3. Histogram of the estimated pdf of the download time (T_i) with mesh architecture and 10^4 peers.

Figure 3 shows the histogram of the estimated pdf of T_i for a network with 10^4 peers and the input pdf of Table I. The distribution process follows a mesh-based architecture. All peers end in at most one round.

Although distributions like the one depicted in Fig. 3 are the prime output of our solution tool, in the following we show only aggregate results, that are more compact, and still convey the fundamental meaning of the results we obtained.

A. Tree Based Distribution Processes

We start by evaluating the influence of the neighbor set size in the overlay network on the delay in the content distribution network (see Fig. 4).

We consider a uniform random connectivity among peers. If the neighbor set is small (4 neighbors), the mean download time grows for an increasing number of nodes much faster than for the case where the number of neighbors is equal to $\mathcal{N}$. However, if a peer has neighbors uniformly chosen from the peer set and if the neighbor set is large enough the performance is independent of the size of the neighbor set, and $\overline{T}$ grows as $\log(\mathcal{N})$ as we expect in a tree topology.

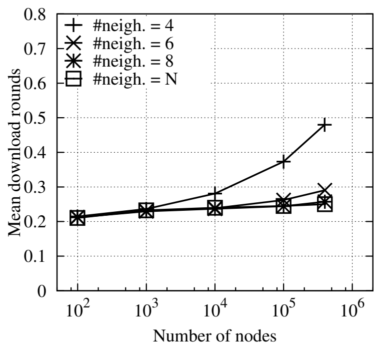

Fig. 4. $\overline{T}$ varying the number of neighbors for each peer (outdegree 1-4), for increasing number of peers.

These observations are valid independently from the outdegree constraints (results are not shown here) and the kind of process (unbalanced, uneven), i.e., trees of the same type with the same outdegree bounds obtain the same performance independent of the size of the neighbor set.

Fig. 5 focuses on the comparison between unbalanced and uneven trees. We study the influence of the different constraints, such as $k^{\max}$ and $k^{\min}$. The results are plotted as a function of $\mathcal{N}$. The poorer results for unbalanced trees are due to the bounds on the $d^{(i)}$. Slow peers, especially those close to the root, impose their rate on the whole subtree, independently of the bandwidth of the peers in the subtree. In the case of uneven trees, slow peers close to the root have no time to start to upload, since the time it takes to become eligible is larger than the time it takes the fast peers to reach, at different levels, all the other peers. This increase of performance for the uneven tree comes at a cost of a greater step distance.

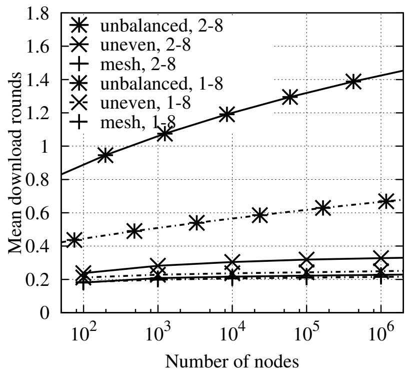

Fig. 5. $\overline{T}$ for unbalanced, uneven trees and mesh with different outdegrees as a function of $\mathcal{N}$.

Another parameter that has a major impact on the performance is the minimum outdegree: imposing a minimum number of children equal to 2 means that a slow peer will divide its low upload bandwidth by 2, which, in the case of an unbalanced tree, will effect the entire subtree.

B. Mesh Based Distribution Processes

Fig. 5 reports also results for meshes. As already observed $\overline{T}$ increases logarithmically with $\mathcal{N}$. A mesh architecture can fully exploit the spare bandwidth of the peers, especially with a minimum outdegree of 2, thanks to multiple connections to other peers. Such an architecture is also more resilient to peer failures.

In order to understand better the performance of the different architectures, we show in Fig. 6 the Cumulative Distribution Function (CDF) of the download times in a network with 10^5 peers.

The interesting point is indeed more related to the capability of uneven trees to efficiently use the different bandwidths in a non-homogeneous network, leading to results that may look counter-intuitive when thinking about the more familiar case of tree and mesh architectures for homogeneous networks.

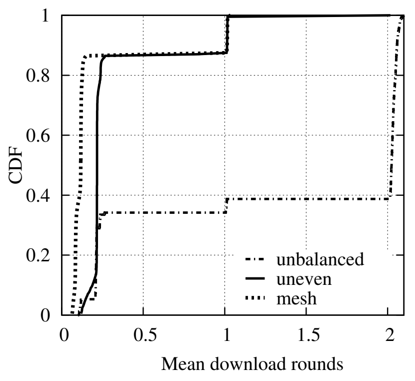

Fig. 6. Comparison between unbalanced tree, uneven tree and mesh for a community of 10^5 peers (outdegree 2-8).

The benefits of a mesh architecture can be also observed looking at the wasted upload bandwidth. This is a measure of how much of the peer upload bandwidth is unused when the peer is involved in the distribution process. Table II shows the percentage of wasted bandwidth when using a tree (uneven) and a mesh for different outdegrees and community size. The waste reduction in the mesh is clearly due to the fact that the upload bandwidth of the leaves is used efficiently.

TABLE II
COMPARISON OF TREES AND MESHES.

		Upload Wasted Bandwidth	
Outdegree	#peers	Uneven Tree	Mesh
1 - 8	10^5	46.9%	13.3%
1 - 8	10^6	47.5%	13.1%
2 - 8	10^5	66.2%	26.8%
2 - 8	10^6	68.9%	29.3%

V. CONCLUSIONS

Distribution systems are based on tree or mesh topologies; however very few works addressed fundamental features of the topologies in presence of non-regular building rules (as in any real world protocol) and network heterogeneities.

The contribution of this paper lies in the analysis of fundamental features, a contribution enabled by an extremely efficient numerical solver of the system evolution equations. We can obtain results for meshes with a million peers and for trees with several tens of millions of peers within a few hours of a standard PC CPU time.

For tree-based topologies, we show that in case of bandwidth heterogeneity it is quite important for the overall performance and efficiency of the tree that the peers with low bandwidth are not at the *root of large subtrees* of peers, as this will lead to poor efficiency and higher download times. Results show that uneven trees perform much better than unbalanced trees since uneven trees succeed at placing slow peers mainly at the leaves. Another parameter that has not been considered before is the minimum peer outdegree. Allowing for a minimum peer outdegree of 1 as compared to 2 can cut the download time into half, because it better exploits the peer download bandwidth.

The methodology and the tool we have developed can be extended to explore scenarios where the available bandwidth of a peer varies over time and where peers can leave and join dynamically. We plan these extensions as a future work together with the exploration of the effect of selfish peers.

ACKNOWLEDGMENT

This work has been partly supported by the European Union under the E-NEXT project FP6-506869.

REFERENCES

[1] B. Cohen, "Incentives build robustness in BitTorrent," May 2003. Available: `http://www.bittorrent.com/documentation.html`

[2] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling Peer-Peer File Sharing Systems," in *Proc. IEEE INFOCOM 2003*, San Francisco, California, USA, Mar. 2003.

[3] X. Yang and G. de Veciana, "Service Capacity of Peer-to-Peer Networks," in *Proc. IEEE INFOCOM 2004*, Hong Kong, Mar. 2004.

[4] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *Proc. ACM SIGCOMM 2004*, Portland, OR, Sept. 2004.

[5] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, S. Sahu "Scalability of Reliable Group Communication Using Overlays," in *Proc. IEEE INFOCOM 2004*, Hong Kong, Mar. 2004.

[6] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proc. Multimedia Computing and Networking 2002*, San Jose, CA, USA, Jan. 2002.

[7] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," in *Proc. of the 3rd Usenix Symposium on Internet Technologies & Systems (USITS)*, Mar. 2001.

[8] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: Highbandwidth Multicast in a Cooperative Environment," in *Proc. ACM Symposium on Operating Systems Principles (SOSP 03)*, The Sagamore, New York, USA, Oct. 2003.

[9] E. W. Biersack, P. Rodriguez, and P. Felber, "Performance Analysis of Peer-to-Peer Networks for File Distribution" in *Proc. 5th International Workshop on Quality of Future Internet Services (QofIS'04)*, Barcelona, Spain, Sept. 2004.

[10] D. Carra, and R. Lo Cigno, "Stochastic Analysis of Chain Based File Distribution Architectures with Heterogeneous Peers," in *Proc. WCW 2005 (WCW 2005)*, Sophia Antipolis, Nice, France, Sept. 2005.

[11] S. M. Ross, "Introduction to Probability and Statistics for Engineers and Scientists," Academic Press, 8th edition, Dec. 2002.

[12] D. Carra, R. Lo Cigno, and E. W. Biersack, "Fast Stochastic Exploration of P2P File Distribution Architectures," Technical Report DIT-06-014, Univ. of Trento, Feb. 2006. Available: `http://www.dit.unitn.it/locigno/preprints/DIT-06-014.pdf`