UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo — Trento (Italy), Via Sommarive 14
http://dit.unitn.it/

ANALYSIS OF CONTENT DISTRIBUTION ARCHITECTURES IN P2P SYSTEMS

D. Carra, R. Lo Cigno and E.W. Biersack

November 2005 – Ver. 1.0

Technical Report # DIT-05-079

Analysis of Content Distribution Architectures in P2P Systems

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*— This paper takes a novel perspective to P2P networking. Can a P2P system be used for the distribution of files with time-critical data such as relevant software patches or virus footprints updates?**

We examine and compare different distribution architectures based on linear and tree topologies built on top of the P2P overlay, including in the analysis the presence of heterogeneous bandwidths, both symmetric and asymmetric access links.

We propose an analytical solution of the distribution process that not only yields the mean download time but also the distribution of the download times. We validate the analytical model against a Monte Carlo based numerical solution, which can also be used to analyze scenarios where correlation and dynamic behavior make the theoretical analysis too approximate.

The insights we gain are used to devise modifications of the distribution strategies that achieve good performance even when slow access links and incomplete knowledge jeopardize the fast delivery of the content. Indeed, with proper (yet simple) mechanisms the average completion times achieved are close to the minimal (lowest possible) ones, which indicates that the P2P paradigm can be used for delivering time-critical data.

Finally the presence of non-cooperative peers is analyzed, assessing their impact for different scenarios.

I. Introduction

Peer-to-peer systems, in which peer computers form an overlay network and share their resources (storage, CPU, bandwidth), have attracted a lot of interest lately. They provide a great potential for building cooperative networks that are self-organizing, efficient, and scalable [1].

Research in peer-to-peer networks has so far mainly focused on content storage and lookup and overlay formation, but few efforts have been spent to analyze and optimize the distribution phase. By capitalizing on the bandwidth of peer nodes, cooperative architectures offer great potential for addressing one of the most challenging issue of today's Internet: the cost-effective distribution of bandwidth intensive content to thousands of simultaneous users both Internet-wide and in private networks.

Cooperative content distribution networks are inherently self-scalable, in that the capacity of the system increases as more peers arrive: each new peer requests service from, but also provides service to, the other peers. The network can spontaneously adapt to the demand by taking advantage of the resources provided by every peer, but the organization of the distribution process, the network characteristics, and the peer behavior can greatly influence the overall performance.

In this paper we focus on the distribution of time-critical data, in the form of a single file with size F that must be delivered to all peers in the network. We assume a BitTorrent-like distribution protocol [2], where the file is broken up into C pieces called "chunks," and peers that have received a chunk are able to upload it to other peers.

The time it takes to download the file to all peers depends on how the chunks propagate between peers, which is referred to as peer organization strategy, or *distribution architecture*. We consider the distribution architectures identified in [3], briefly summarized in Sect. II, and derive a stochastic model for the download performance in presence of peers with heterogeneous bandwidth. One of the aims of the analysis is to obtain insights that can help in designing more efficient distribution protocols. The backbone network is considered ideal, with unlimited bandwidth. The impact of backbone bottlenecks is left for future study.

The main contributions of the paper are the following.

We derive an approximated analytical model that yields the stochastic distribution of the file delivery performance as a function of the distribution of the peer bandwidth. The model allows to assess the impact of slow peers on the delivery and gives enough insight in the problem to devise dynamic distribution strategies to overcome the impact of slow peers. In tree based architectures, the model yields a lower bound on performance, thus it is suited for design and dimensioning.

A simple and effective Monte Carlo technique is devised to validate the model results and to analyze distribution architectures where the tree degree can vary dynamically based on locally available resources. In particular, we analyze the effect of bounds of the node outdegree (the tree degree local to a given node) on global performance.

The same Monte Carlo technique is used to assess the performance of the distribution in presence of selfish or malfunctioning peers as a function of the distribution architecture. Results shows that selfish peers have no great impact and the results on performance we obtain considering no selfish peer still hold.

The paper is organized as follows. Sect. II introduces the distribution architectures we analyze in the paper. Sect. III describes the analytical model and the approximations introduced. In Sect. IV we introduce the Monte Carlo simulator and validate the analytical model. Sects. V and VI discuss results in

different networking scenarios and in presence of selfish peers. Sect, VII is devoted to discuss related works, and Sect. VIII ends the paper with some additional discussions.

II. Distribution Architectures

The main focus is the analysis of fundamental distribution architectures. Most of the configurations used in distribution processes are built with two basic structures: chains and trees.

A. Chains

The simplest architecture that can be defined is a single chain, where each node downloads from exactly one node and uploads to exactly one node. The distribution process generates different, parallel, independent chains, and a peer can be a node in one chain only. Even if this architecture seems too simple to be of practical interest, it serves as a basis for comparative analysis. Moreover, it can be used as simple building block that can form more complex distribution architectures, and in some cases it might suffice for the application purposes. It is worth noticing that chains are fair in the sense that each peer (excluded the last one in the chain) uploads exactly the same amount of information it downloads.

B. Tree

In tree structures, each node downloads from exactly one node and uploads to k nodes, where k is the *outdegree* of the tree. The choice of k has two opposite effects: On the one side, by increasing k it is possible to reach the leaf nodes in fewer steps. On the other side, the bandwidth used in transferring the content from a peer to its children is divided among k peers. Therefore, increasing k will increase the total download time from one level of the tree to the next one. Considering a theoretical model where all peers have the same bandwidth and the tree is balanced, the binary tree is the best compromise between these two effects [3].

Tree based distribution architectures are an efficient way to distribute contents, given performance and simplicity. Nevertheless they suffer from two main shortcomings: (i) only interior nodes upload to other nodes, whereas leaf nodes do not upload at all and (ii) nodes use only part their download bandwidth (if the tree is balanced and peers are homogeneous, each node receives $1/k$ of the uploading bandwidth of its parent node). The first aspect results in *unfairness* among nodes: some nodes upload k times the amount of data they receive, while other nodes (leafs) only receive data. Since the number of leaf nodes is greater than the number of interior nodes[1], this also implies a great waste of upload bandwidth, since the majority of the nodes do not contribute to the distribution of the content. The second aspect implies that the time necessary to complete the download is k times larger than the time it would take if the file were downloaded using the full download bandwidth of the parent node (this is strictly true for the homogeneous case, both with symmetric and asymmetric bandwidths).

[1]In a balanced tree with l levels and outdegree k, $k \geq 2$, the number of leaf nodes and interior nodes is respectively k^l and $\frac{k^l-1}{k-1}$.

C. PTree

The overall performance of a tree based architecture is significantly improved by capitalizing the unused upload bandwidth of the leaf nodes and the unused download bandwidth of internal peers. A solution in this direction is given by SplitStream [4]. The SplitStream architecture was defined for streaming services in a structured P2P system (based on Pastry), and it was extended to file distribution processes without any particular structure of the overlay network in [3]; in this generalized form, the architecture is called PTree.

In PTree the content is divided into r different stripes and each stripe is distributed by a different tree. A peer participates in all distribution trees, but with different roles. Precisely, a peer is an interior node in at most one tree, while it is a leaf node in the remaining $r-1$ trees. In this situation, when the node is interior, it uploads one stripe of the file k times, while it is receiving the other $r-1$ stripes in parallel since it is a leaf node of the other trees. If we choose to divide the file in exactly k stripes, i.e., $r = k$, we obtain complete fairness (see Fig. 1): each peer uploads the same amount of data it receives. There is only one exception: independent of the outdegree k, there will always be one peer that is leaf node in all r trees (the shaded circle in Fig. 1).

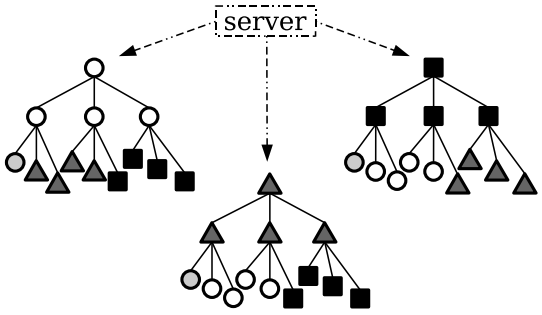

Fig. 1. Example of a PTree distribution architecture with 13 nodes and 3 stripes, each node appears once in every distribution tree.

If we choose $r < k$, we obtain a situation where some peers are interior nodes of one tree and leaves of the other trees and they upload k/r times what they receive, whereas other peers are only leaf nodes, so they only receive the whole file. This sort of mild unfairness can be considered acceptable, since nodes behind firewalls and NAT devices can not participate actively in the distribution process. We do not consider the case $r > k$: the number of leaf nodes of one tree is not sufficient to cover the interior nodes of the other $r-1$ trees, and a subset of nodes must act as interior node in more than one tree. We consider only the case $r \leq k$.

D. Prior Work

In [3] the performance of file distribution on the above architectures was tackled. Assuming a BitTorrent-like protocol and a homogeneous case, where all peers have the same (symmetric) bandwidth, an upper bound on the number of peers served within an interval time t was derived. The impact of basic parameters, such as the outdegree of tree architectures

or PTree, and the number of chunks in which the file is divided was assessed.

The strongest result is that in this ideal case PTree is able to offer download times close to F/b, that is the time it would take a single node to download the file via unicast. This result holds for any large number of peers in the network, provided that the number of levels of the trees that compose the PTree architecture is smaller than the number of chunks.

An interesting result regards the simple chain based architecture: when the number of chunks C is comparable to the number of peers N, peers stay engaged most of the time and the performances are equivalent to those obtained with PTree. The pivotal point up to which a simple chain based architecture is comparable to PTree is around $N = C$.

In [5] the impact of bandwidth heterogeneity in simple chain based architectures is considered. It was shown that if *complete knowledge* of the system is available and all the peers cooperate the heterogeneity is not detrimental and fast peers effectively help slow ones without being affected. Without complete knowledge, the deterministic choice (and consequently also the analysis) of peers is not possible and a basic stochastic model was introduced. Under these assumptions, the download time as a function of the percentage of slow peers and the number of chunks were studied.

The modeling technique we present here is based on these preliminary works.

III. The Analytical Model

Consider a scenario where peers have different symmetric access link bandwidths (we discuss implications of asymmetric bandwidths in Sect. VI). There is only one server in the system with bandwidth at least equal to the highest peer bandwidth. All peers are independent, so we can describe the system through a random variable $b_{\text{p\_i}}$, the bandwidth of peer i, having a known density, which is identically distributed for all peers. The density function of $b_{\text{p\_i}}$ summarizes the fact that peers in the network might dedicate only part of the bandwidth for file distribution and also the fact that there could be peers with different access technologies. The bandwidth of a peer stays constant during the whole file transfer.

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 F; the time needed to download the complete file at the lowest bandwidth in the network is referred to as $T_R = \frac{F}{\min(b_{\text{p\_i}})}$ and is also called *one round*,.

Initially, we suppose that peers remain on line till the end of the distribution process. The relaxation of this hypothesis is discussed in Sect. VI. 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 access link are present.

The delivery process is distributed and the topology (chain, tree, ...) is built step-by-step. Each peer has a (possibly partial) list of the peers involved in the distribution process and contacts a node in the list randomly, checking that it has not yet been contacted. The optimization of this process is beyond the scope of this contribution.

A. Chain based Architecture

The basic idea in our analytical evaluation is to start from the probability density function (pdf) of $b_{\text{p\_i}}$, representing the bandwidth of peer i, and compute how the actual transfer bandwidth evolves along the delivery process due to interaction of the nodes.

We consider a single chain; due to chain independence, the result can be extended easily to multiple chains.

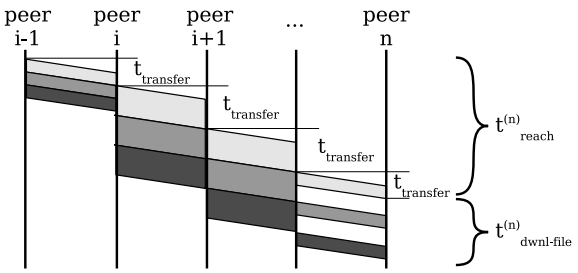

Fig. 2. File transfer over a single chain. All the peers have the same bandwidth except peer $i+1$ that has a smaller one.

Given a peer n, the total download time can be divided into two terms: the time necessary to receive the first chunk and the time necessary to upload the remaining $C-1$ chunks to that peer (see Fig. 2). We have

$$t^{(n)}_{\text{total}} = t^{(n)}_{\text{reach}} + t^{(n)}_{\text{dwnl-file}} \tag{1}$$

We define the *chunk transfer time* t_{C} as the time necessary to transfer a single chunk between two peers. The chunk transfer time is equal to the size F/C of the chunk, divided by the *transfer bandwidth*, i.e., the bandwidth used to transmit the chunk between two peers. Let b_i be the transfer bandwidth between peers $i-1$ and i, we have $b_i = \min(b_{\text{p\_(i-1)}}, b_{\text{p\_i}})$.

The term $t^{(n)}_{\text{reach}}$ is the sum of all the chunk transfer times:

$$t^{(n)}_{\text{reach}} = t^{(n-1)}_{\text{reach}} + \frac{F}{Cb_n} = \frac{F}{C}\sum_{i=1}^{n}\frac{1}{b_i} \tag{2}$$

Step n can be derived from $t^{(n-1)}_{\text{reach}}$ and the characteristics of peer n.

The random variables in (1) and (2) are correlated. However, the correlation is weak in both cases and we assume independence between $t^{(n)}_{\text{reach}}$ and $t^{(n)}_{\text{dwnl-file}}$ as well as between all b_i, to derive an approximated solution via convolution of the pdfs using an iterative procedure (the validation of this approximation is reported in Sect. IV-B).

Given the pdf of $t^{(n-1)}_{\text{reach}}$ and the pdf of $1/b_n$, that can be simply found considering that $b_n = \min(b_{\text{p\_(n-1)}}, b_{\text{p\_n}})$, we can obtain the pdf of $t^{(n)}_{\text{reach}}$ by convolution.

The term $t^{(n)}_{\text{dwnl-file}}$ depends on the interval time between chunks, which is constant since the bandwidth is constant. The time necessary to download the file, i.e., to download the $C-1$ chunks once the first chunk has reached the peer, is $C-1$ times the interval between chunks. This interval is given by the lowest peer bandwidth encountered, i.e., it is the maximum of the value $\frac{F}{Cb_{\text{p\_i}}}$.

$$\begin{aligned} t^{(n)}_{\text{dwnl-file}} &= \max\left(t^{(n-1)}_{\text{dwnl-file}}, (C-1)\frac{F}{Cb_{\text{p\_n}}}\right) \\ &= (C-1)\max\left(\frac{F}{Cb_{\text{p\_1}}}, \frac{F}{Cb_{\text{p\_2}}}, \ldots, \frac{F}{Cb_{\text{p\_n}}}\right) \quad (3) \end{aligned}$$

We emphasize that step n can be found iteratively from step $n-1$ and peer n; since peers are independent (no assumptions or approximations here), we can find the pdf of $t^{(n)}_{\text{dwnl-file}}$ from the pdfs of $t^{(n-1)}_{\text{dwnl-file}}$ and $b_{\text{p\_i}}$.

With the complete pdf of the download time at each step we can analyze in detail the behavior of the system with different input pdfs. Examining how the pdf evolves along the path gives insight on the dynamics that mainly influence the distribution process. Besides the mean download time, we can observe the percentage of peers that complete at different intervals: for instance, when the file distribution process has time constraints, it is important to know the percentage of peers reached within a certain time bound.

Multiple chains in parallel are simply analyzed by staggering each of them in time by $F/\max(b_{\text{p\_i}})$.

B. Tree Based Architectures

The analytical model defined in the previous section can be extended to tree architectures. Considering a tree (for instance, a binary tree) and following a sample-path from the root to a leaf we obtain a chain, as pictorially represented by the black nodes in Fig. 3. We can apply the same analysis technique used for chains to the tree case, since, in a stochastic sense, the node in the path at level j of the tree is representative of the whole level.

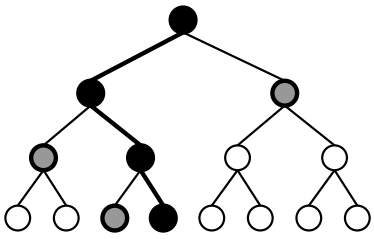

Fig. 3. Example of a sample path in a binary tree; black nodes are in the sample path, gray nodes are those that influence the computation of b_j.

In each step we analyze all the children of a node and assign the bandwidth of the parent node among children: we define the bandwidth b_j used to calculate the chunk transfer time at the j-th step, as the bandwidth assigned according to the max-min fairness criterion. Given the outdegree and the pdf of $b_{\text{p\_i}}$, the $f_{b_j}(b)$ can be easily computed with combinatorial techniques, analyzing the possible permutations of children bandwidths. Given the pdf $f_{b_j}(b)$ the term $t^{(n)}_{\text{reach}}$ is computed with (2).

Now, the term $t^{(n)}_{\text{dwnl-file}}$ must consider the brother nodes (gray circles in Fig. 3). In fact, considering a path, the interval between chunks not only depends on the bandwidths of the peers belonging to the path, but also on the bandwidths of the peers branching from the path. Due to this dependence, (3) is not valid anymore. Nevertheless, with the approximation of independent transfer times we have

$$\begin{aligned} t^{(n)}_{\text{dwnl-file}} &= \max\left(t^{(n-1)}_{\text{dwnl-file}}, (C-1)\frac{F}{Cb_n}\right) \\ &= (C-1)\max\left(\frac{F}{Cb_1}, \frac{F}{Cb_2}, \ldots, \frac{F}{Cb_n}\right) \quad (4) \end{aligned}$$

Equation 4 still allows for an iterative solution, with the only drawback that we consider twice the bandwidth of each node, which leads to overestimate the transfer time as we can see in the simple example of Fig. 4. We assume a binary tree and let p_F be the probability to have a fast node. The maximum rate in a step is $b_{j\max} = (b_{\text{fast}} - b_{\text{slow}})$, where b_{fast} and b_{slow} are the bandwidths of fast and slow peers respectively. To have two consecutive maximum rate steps we need three fast and two slow nodes involved (left hand side of Fig. 4), so that the probability of this event is $p_{2\text{f}} = p_F^3(1-p_F)^2$. Assuming instead that steps are independent, is equivalent to "duplicating" nodes, as in the right hand side of the figure, which implies that the approximated probability of having two consecutive maximum rate steps is $p'_{2\text{f}} = p_F^4(1-p_F)^2 < p_{2\text{f}}$. Repeating the operation for any number of fast steps yields that the probability of fast (not only the fastest) paths is *underestimated* when we assume independent steps.

On the other hand, to have a slow path we just need a single slow step, which occurs with probability $p_{1\text{f}} = (1-p_F)^3$ and is unchanged by the independence approximation. This simple induction implies that the download time computed assuming independence is an upper bound to the real download time.

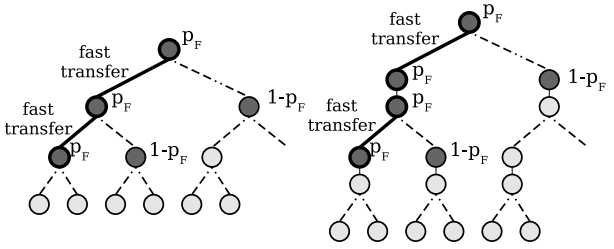

Fig. 4. Real distribution tree (left) and equivalent tree with "duplicated" nodes considering independent steps (right).

The analysis of PTrees is surprisingly trivial, with the assumption that each node gives priority to the download of the stripe for which it is an interior node (say stripe m), and starts downloading the other stripes (for which it is a leaf) only after it finishes downloading stripe m. This assumption is reasonable because the order of stripe downloads does not affect the total download time for a node. Therefore, a node has thus no reason to delay the delivery of the stripe for which

it is an interior node. The upload of stripe m to other peers does not affect the download of the other stripes.

In PTree, every peer is interior node in one tree and leaf node in other $r-1$ trees; this implies that (i) the total download time is dominated by the stripes received as leaf node, and (ii) all peers are equal, i.e., they have the same pdf of the total download time. The rate at which the file arrives at leaf nodes is determined by the slowest bandwidth encountered in the path; the probability that this rate is equal to the lowest bandwidth in the system ($\frac{b_{\text{slow}}}{k}$, since the bandwidth is divided among the k children) increases rapidly with the depth of the tree. We approximate the pdf to a single Dirac's delta, obtaining (we recall that each stripe is F/r bits)

$$t_{\text{total}}^{(l)} = \frac{k}{r}\frac{F}{b_{\text{slow}}} + t_{\text{reach}}^{(l)} \tag{5}$$

where $t_{\text{reach}}^{(l)}$ is the time necessary to reach leaf nodes when the tree depth is l. The term $t_{\text{reach}}^{(l)}$ grows logarithmically with n, number of peers, and we can suppose that is equal to l times the mean chunk transfer time between two peers. If we set $k = r$ we obtain that all peers terminate approximately at $\frac{F}{b_{\text{slow}}} + l \cdot t_{\text{Cslow}}$, where t_{Cslow} is the chunk transfer time for the slowest peer.

C. Results of the Analytical Model

As numerical example we consider two different density functions for the peer bandwidth, summarized in Table I. The distribution in the upper table is taken from [6]. The other one is an example that aims at studying the interaction among high speed peers that reserve a percentage of their bandwidth for other tasks. We assume that the bandwidth available for the file distribution process is uniformly distributed between 80% and 100% of the peer bandwidth.

TABLE I

RATE DISTRIBUTIONS USED IN THE EXAMPLES

Rate distribution A	
Rate	Weight
56 kbit/s	13%
640 kbit/s	23%
1.2 Mbit/s	64%

Rate distribution B		
Rate	Density	Weight
640 kbit/s	Uniform(80%-100%)	30%
2 Mbit/s	Uniform(80%-100%)	70%

When reporting results, we normalize the data such that $T_R = \frac{F}{\min(b_{\text{p\_i}})} = 1$ round. 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 far as $1 \ll C \ll N$, where N is the total number of peers.

Fig. 5 shows how the pdf of $t_{\text{dwnl-file}}$ evolves along different levels of the same tree. In this case we consider a binary tree with input pdf A from Table I. Initially (we show here the 8-th level of the tree) the contribution of the three classes of peers is clear (continuous lines); the maximum download time is 2 rounds since the outdegree is 2. As the number of levels increases (when we reach the 25-th level of the tree), the distribution starts to concentrate on the maximum download time (dashed line), because the minimum bandwidth dominates.

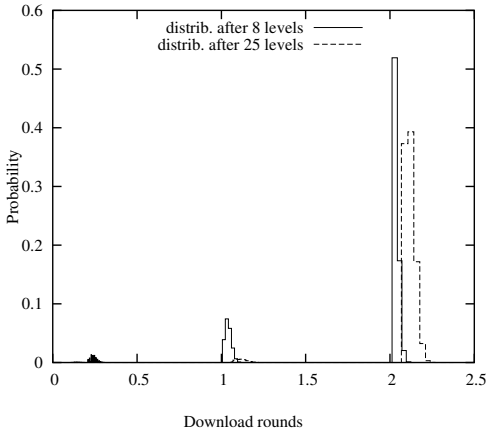

Fig. 5. Evolution of the pdfs of the download time for two different tree heights.

Fig. 6 shows results of chain architecture and the tree with outdegree 2 and 3, using the density A of Table I. We compare the heterogeneous case with the homogeneous one: the comparison considers for the homogeneous case an equivalent bandwidth equal to the mean transfer bandwidth of a single step. We do not present here results for PTree for clarity, since its behavior is clear from Fig. 7.

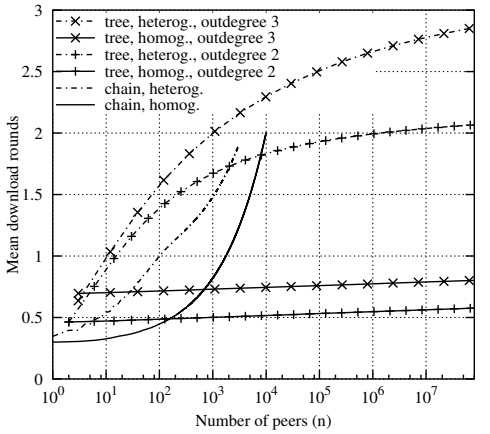

Fig. 6. Mean completion time for a given number of peers: comparison between chain based and tree based architecture, heterogeneous peers (rate distribution A) and homogeneous with average bandwidth.

A detailed analysis of the results for homogeneous case can be found in [3]. Here we report a few essential observations. In a tree architecture, each peer uploads with bandwidth b_{p}/k, so the minimum necessary download time is kF/b_{p}. The time to receive the first chunk grows logarithmically in the number of peers since in a homogeneous environment all the transfers occur with the same rate and the only difference in the total

download time is the time necessary to reach the l-th level.

The performance degradation in case of heterogeneous bandwidth is clearly due to the fact that a slow peer influences all the downstream nodes, regardless of their bandwidths. For instance, in case of tree architecture, the subtree under the slow peer proceeds slowly, so the number of peers that are forced to continue with low rate grows exponentially. The results of these dynamics are the convergence of the curve to k rounds. After few levels, the total download time of each peer, $kF/b_{\text{p\_i}}$, tends to become $kF/b_{\text{slow}} = k$ rounds. The chain based architecture has the same behavior. In general, the heterogeneous case converges, as the number of peers increases, to the homogeneous case, with the homogeneous bandwidth equal to the minimum bandwidth. As is clear from the analysis of the chains, the download time increases polynomially with the number of peers.

We observe the same behavior with different input pdfs. Fig. 7 presents the results using the density B of Table I. We remark only a slight difference due to higher percentage of slow peers than in the previous case, so the curves converge faster to k rounds. For PTree architecture, results converge to the maximum download time for all peers. Note that results are normalized using the smallest bandwidth in the system: in the two cases (with input pdf A and B) the normalization factors are different and so the absolute times.

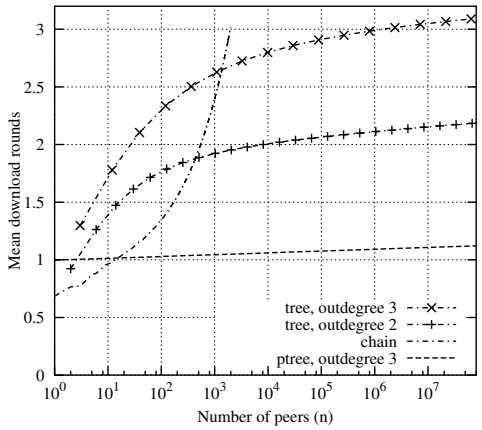

Fig. 7. Mean completion time for a given number of peers: comparison between chain based and tree based architecture (rate distribution B).

From the study of the analytical results it is possible to draw some observations. As the number of peers grows, tree-based architectures are obviously faster than chain based ones, but tree-based architecture pay an initial cost due to the multiplication of the download time by the outdegree k. Therefore, initially, short chains are faster than small trees. Biersack et al. [3] demonstrated that the cross point between chain and binary tree in the homogeneous case is a function of the number of chunks C. In the heterogeneous case, we can observe that this cross-over point occurs for a number of peers that is one order of magnitude greater than in the homogeneous case. We conclude that short chains are preferable up to roughly C peers.

IV. Monte Carlo Numerical Solution

In order to evaluate the impact of the independence assumptions made in solving the analytical model, we developed a simulator that takes into account all correlations in the different distribution architectures . The simulator can also handle dynamic distribution Trees and PTrees and other cases discussed later and can be extended to generic meshes. For the simulator we use the same general assumptions made for the analytical model (Sect. III).

A. Simulator Description

The simulator performs essentially a Monte Carlo analysis of the solution space given a distribution architecture. It takes as input any discrete pdf of peer bandwidths and builds step by step a sample path. By performing the building process multiple times, it derives a histogram of the pdf of the relevant performance figures.

Consider for simplicity a single chain. Let n be the number of nodes in the chain and w the number of different possible bandwidths (for example, if we consider rate distribution A in Table I, we have $w = 3$). Then the number of possible chains is $O(w^n)$. The simulator randomly samples the solution space and builds the histogram of the transfer rate at each step. We terminate sampling process when the mean and the 90-percentile of the total download time reaches a 99% confidence level for an interval $\pm 10\%$ of the point estimate. We use the batch means technique with repeated independent simulations.

The simulations are fast: indeed, even if the solution space is huge, the transfer rate is dominated by the lowest bandwidth encountered along the path, and the histogram converges rapidly. Moreover, if the bandwidth of the peer are sufficiently large, the tail of the distribution is limited and its upper bound can be easily estimated.

In tree architectures we use a single path in the tree as we did for the analysis. At each step down the tree, a number of children nodes equal to the outdegree of the parent tree are randomly chosen, and the bandwidth is divided among the different children according to the max-min fairness criterion. The following step in the path is selected choosing randomly one of the children nodes and repeating the process.

Interestingly the space to be explored in the tree case is $O(w^{n'})$, with $n' = k \log_k(n)$, thus it is even smaller than for the chain architecture. This also holds for PTrees.

B. Comparison with the Analysis

Fig. 8 compares the results of the analytical model with the ones of the simulator, using the rate distributions A and B of Table I. The upper plot refers to distribution A, and the lower one to distribution B. In the chain and PTree architectures, the analytical model fits the simulation experiments almost exactly.

Considering the tree architecture, as discussed in Sect. III-C, the analysis yields indeed an upper bound of the actual average download time, as is confirmed by the simulation results. It is interesting to notice that the bound becomes tighter as the

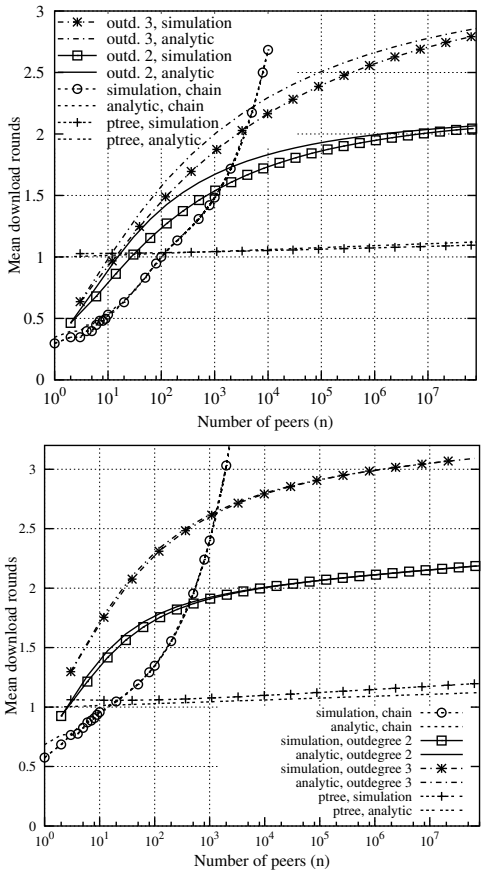

Fig. 8. Comparison between analysis and simulations with different architectures; the upper plot refers to rate distribution A and the lower one to B.

number of peer increases and as the fraction of slow peer increases (lower plot). This is easily understood considering that the overestimation comes from underestimating the weight of fast paths down the tree and thus disappears as the probability of fast paths tends to zero.

In the following sections we discuss some interesting results in cases that do not lend themselves to an easy theoretical analysis, and all results presented are obtained using the Monte Carlo simulator.

V. Improving the Distribution Architecture

A. Analysis of Hybrid Architectures

With tree architectures we have a degree of freedom in the design process: the outdegree. In the homogeneous case, the decision of the outdegree is a classical optimization problem. In a heterogeneous environment, when it is not possible to know the bandwidth of the children nodes, the optimal outdegree does not exist. The best solution is to let the outdegree to *dynamically adapt* to the changing conditions.

With a dynamic outdegree, peers need a criterion to set the outdegree. There are essentially two cases where increasing the outdegree is beneficial:

- The selected children are not able to use all the parent peer's upload bandwidth;
- A peer is downloading from a parent with a rate lower than its upload rate.

The first case is self explanatory. The second case can be explained as follows. If a peer has a bandwidth b and a number of children equal to k, it takes $t_u = \frac{\text{chunk\_size}}{b/k} = k\frac{\text{chunk\_size}}{b}$ to distribute one chunk to all its children. Let t_δ be the download time of a chunk; if $t_\delta > t_u$ the peer can increase the number of children k until reaching $t_u \simeq t_\delta$. The value of t_δ can be estimated during the download process, so that a peer can increase its own outdegree without the risk of becoming a bottleneck.

Fig. 9 shows the mean download time, given the mean number of peers. We consider the mean number of peers since we measure the mean outdegree at each level and then we accordingly calculate the mean number of peers in each level. The bandwidth distribution is A, and we consider different upper limits to the outdegree.

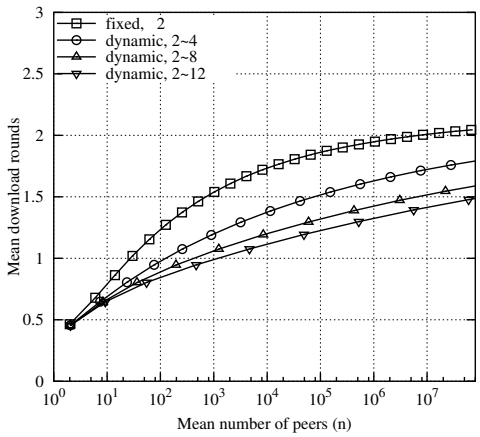

Fig. 9. Mean completion time for a given number of peers in a tree based architecture with dynamic outdegree; rate distribution A.

As the maximum number of allowed children increases, the mean download time decreases. The explanation of such results can be found analyzing the cumulative distribution of the completion time. We set the same number of reached nodes and we compare the total download time of trees with different variable outdegree. Fig. 10 shows the percentage of completed peers as a function of time for $n = 10^7$.

Allowing the outdegree to increase, enables more and more peers to be reached in a given level. Moreover it enhances the probability that one of the children is fast. This is reflected by a larger fraction of peers ending the transfer early (thus reducing the average). However, most of the peers still need 2 rounds to finish, and the gain we have as the maximum outdegree increases becomes progressively smaller.

B. Changing the Minimum Outdegree

Fig. 10 highlights the fact that the performance is dominated by peers that terminate at time 2 rounds. In a tree with a minimum outdegree $k_{\min}$, the minimum time necessary for slow peers to download the file is $k_{\min}$ rounds. Recall the results obtained in Sect. III-C: short chains are faster than

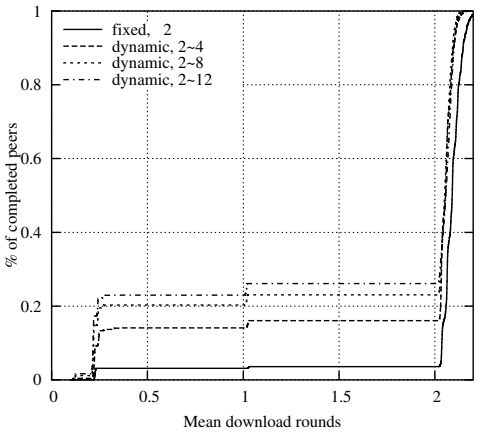

Fig. 10. Percentage of completed peers at a given time in a tree based architecture with dynamic outdegree; rate distribution A.

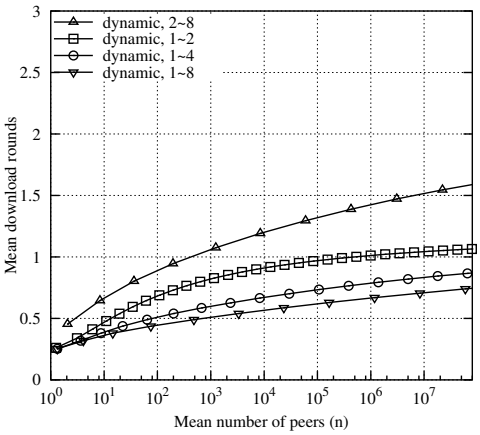

Fig. 12. Mean completion time for a given number of peers in a tree based architecture, with minimum outdegree 1; rate distribution A.

small trees. If we combine chains and trees, with short chains joining different parts of the tree, when the $t_{\text{dwnl-file}}$ via a tree would be larger than 1 round, we obtain a hybrid distribution architecture that should be faster than trees. This is readily obtained allowing a lower outdegree equal to 1. Fig. 11 shows an example of tree evolution with two classes of peers.

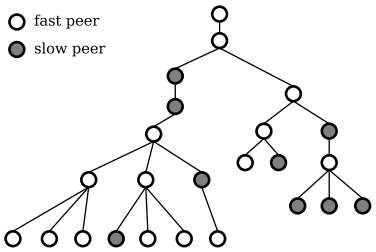

Fig. 11. Example of evolution of fast hybrid tree.

Fig. 12 shows the performance obtained with a minimum outdegree equal to 1, compared to a minimum outdegree 2, using the input rate distribution A.

C. PTree with dynamic outdegree

So far, we have discussed only briefly the performance of PTree, which is however the best performing architecture, since it fully exploits the resources of all peers. On the other hand, the PTree architecture is also the most sophisticated one and can be fragile in case of uncertainty and dynamic behaviors. For example, if the tree grows dynamically and becomes unbalanced, it may happen that in one of the sub-trees there are not enough internal nodes to serve the other peers.

The PTree architecture has two degrees of freedom: the outdegree k and the number of stripes r a file is divided up. Note that the number of stripes is not related to the number of chunks C. In fact, usually the number of chunks is of the order of hundreds to thousands, while the number of stripes can be in the range of 3-8. Each stripe then contains C/r chunks. For simplicity, we consider that C is a multiple of r.

With homogeneous nodes and a fixed outdegree k, PTree allows to receive in parallel r stripes with bandwidth b_{peer}/k each, obtaining a download time equal to $\frac{k}{r}\frac{F}{b_{\text{peer}}}$. Choosing $k = r$, we obtain a minimum download time $t^{\min}_{\text{dwnl-file}} = \frac{F}{b_{\text{peer}}}$, that is the time it takes a node to download the file via unicast communication, hence the optimal performance.

When we introduce heterogeneity, this equilibrium is broken. If a single stripe of the file is downloaded at a smaller rate, say b_{slow}, the whole performance is dominated by the download of this stripe. Indeed, the peer terminates the download only when this stripe is completly received, regardless of the number of stripes that download faster.

Additionally, when introducing a dynamic outdegree, in particular with a minimum outdegree equal to 1, we have to guarantee a minimum level of fairness and, at the same time, that there are enough leaf nodes in any tree to take the role of interior nodes in the remaining $r-1$ trees. In Sect. II-C we stated that $k \geq r$. We have verified empirically by simulation that a value of $k = 1.5r$ is sufficient to guarantee that the distribution process can take place without problems for any reasonable distribution of $b_{\text{p\_i}}$. The value of 1.5 is the ratio of data uploaded to data downloaded, which is in any case better than in a Tree architecture, where half of the peers do not upload at all.

Fig. 13 shows the performance of PTree with $r = 3$ and rate distribution A. Decreasing the minimum outdegree, decreases the mean download time as expected. Surprisingly, the results are only marginally influenced by the maximum outdegree, provided that the maximum outdegree is sufficiently large to ensure a mean outdegree greater than the number of stripes. Still, having a higher outdegree has the advantage of reaching a larger number of nodes for a given number of levels in the tree structure.

Fig. 14 shows the cumulative distribution of the completion time of the peers. Most of the peers terminate roughly at the same time, as they would in a PTree with only fast peers while the 13% of slow peers terminate at time equal to 1 round, which is their best possible performance.

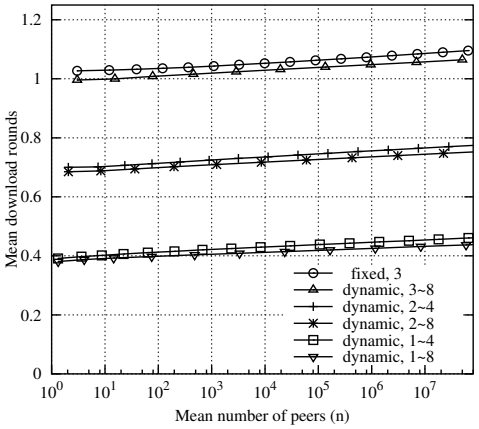

Fig. 13. PTree performance with different dynamic outdegrees; $r = 3$.

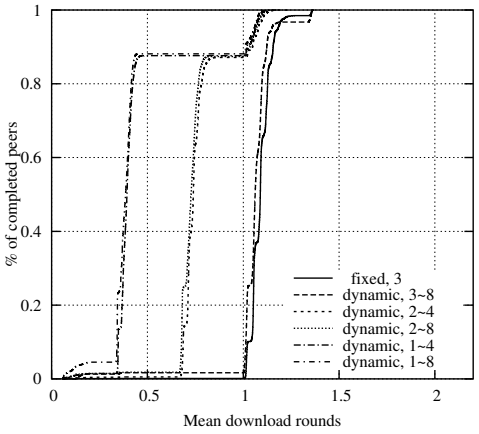

Fig. 14. Percentage of completed peers at a given time in PTree architecture with dynamic outdegree; rate distribution A.

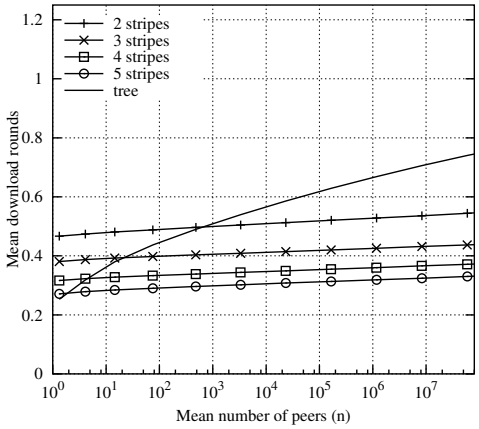

Fig. 15. Tree vs. PTree performance with different number of stripes, the outdegree for all the configurations is 1–8.

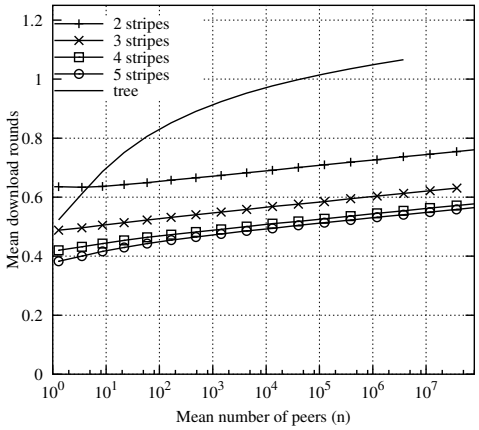

Fig. 16. PTree performance with different number of stripes, rate distribution B (the outdegree for all the configurations is 1–8).

Since performances are not influenced by the maximum outdegree and the optimal minimum outdegree is 1, the last parameter that can be tuned is the number of stripes r. Fig. 15 shows the total download time when the file is divided in different stripes. In all cases, we use a tree with dynamic outdegree in a range of $1 - 8$. As expected, the performance improves with increasing number of stripes. The limit to this gain is given by the ratio between the bandwidth of the fast nodes and slow nodes: we can obtain an improvement as long as the bandwidth of the fastest nodes is r times greater than the one of the slowest node, i.e., it can accept in parallel all the r stripes (one as interior node, the other as leaf node).

The absolute values of the performances depends also on the specific rate distribution. However, Fig. 16, where rate distribution B is used, shows that the relative merit of the distribution architecture as a function of parameters remains unchanged.

VI. Additional Insights

A. Asymmetric Access Bandwidth

Throughout the analysis of the architectures, we have assumed *symmetric* bandwidth for upload and download. Recently, asymmetric broadband access links, like ADSL, have become very popular, and assessing how ADSL affects the download time is important. As a general consideration, we observe that a peer with asymmetric bandwidth can increase its own download performance, but cannot increase the performance of the whole distribution process that leverages on the upload bandwidths.

TABLE II
RATE DISTRIBUTION WITH ASYMMETRIC BANDWIDTHS

Rate Downlink	Rate Uplink	Weight
640 kbit/s	128 kbit/s	20%
2 Mbit/s	2 Mbit/s	80%

Consider the pdf of the bandwith as given in Table II, where one class has ADSL access. We compare the results with the case when the ADSL class has symmetric bandwidth equal to the downlink and uplink bandwidth of the ADSL respectively. Fig. 17 shows results for the Tree architecture. The performance with asymmetric peers is equivalent to the one with symmetric peers with minimum bandwidth.

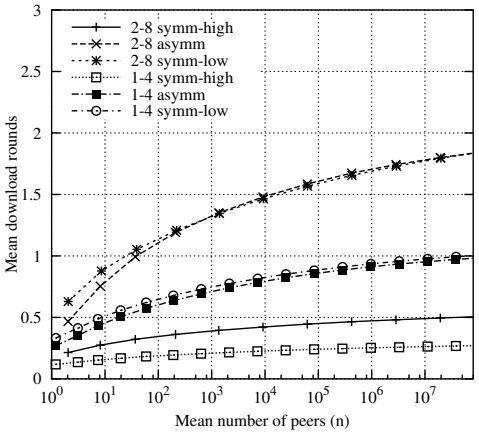

Fig. 17. Mean download time with ADSL (tree architecture) compared with the symmetric case, where the bandwidth is either set to the minimum or to the maximum of the ADSL bandwidth.

We observe a different behavior with PTree (Fig. 18). The performance of asymmetric peers lies between the two symmetric cases. PTree results depend on parallel downloads, each of the download has a rate equal to the minimum rate in the system. In this case ADSL users are the slowest peers and with symmetric bandwidth they can not exploit parallel downloading. With asymmetric bandwidth, instead, the downloading bandwidth can accept all the incoming downloading and the total download time decreases.

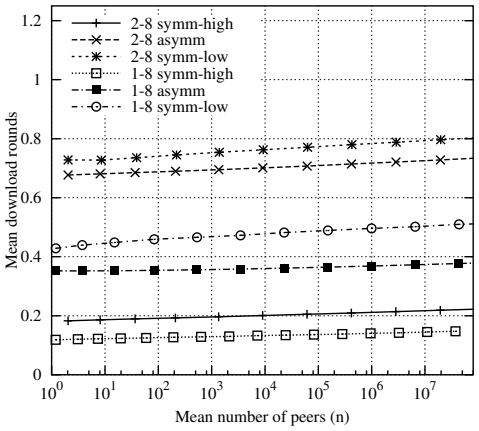

Fig. 18. Mean download time with ADSL (PTree architecture) compared with the symmetric case where the bandwidth is either set to the minimum or to the maximum of the ADSL bandwidth.

B. Selfish Peers

The distribution process relies on peer collaboration: the more altruistic peers are, the faster the distribution. When the file distribution process is done in the global Internet, it has to face with free-riding. Free riders are peers that do not contribute by uploading the file, i.e., they are "selfish," and stall the distribution process.

The impact of selfish peers has been discussed in many papers (e.g., [7] [8]) in generic P2P networks, but not in cooperative content delivery. We assume that each peer has a probability p_{selfish} to be selfish. When we build the sample path in the tree, each step selects its children according to the bandwidths and the outdegree costraints. Since the presence of selfish peers decreases the number of reached peers at each step, in order to maintain the same total number of peers that complete the download, we increase the depth of the tree as p_{selfish} increases.

Fig. 19 shows the results for a tree architecture with different percentages of selfish peers and a maximum outdegree of 8. We can note only slight differences between different curves: in fact the tree structure is weak during the first steps, but, as soon as the depth, and consequently the number of peers, increases, the effect of selfish peers becomes smaller and smaller.

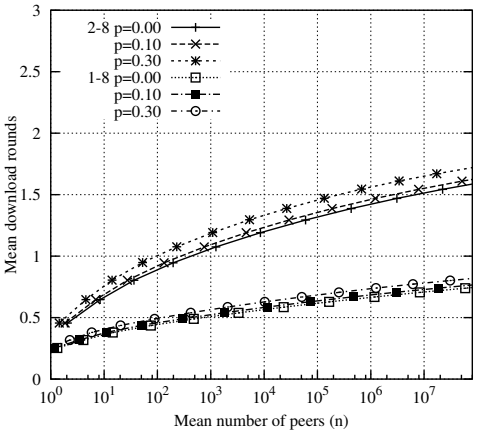

Fig. 19. Mean download time with different percentage of selfish peers (tree architecture): the maximum outdegree is 8.

We obtain the same effect for the PTree architecture (Fig. 20). Since we increase the number of levels in each tree to reach the same number of peers, consequently, as the probability increases, the term $t_{\text{reach}}^{(l)}$ of (5) increases.

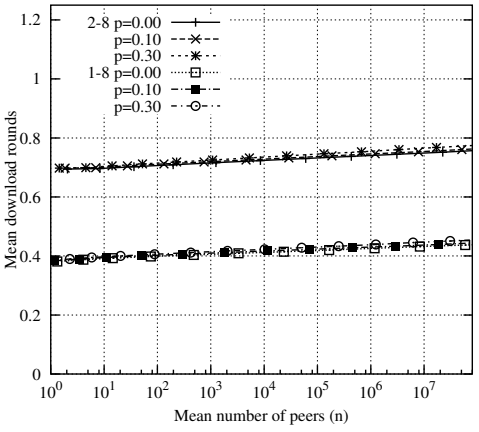

Fig. 20. Mean download time with different percentage of selfish peers (PTree architecture): the maximum outdegree is 8.

C. Mesh-based Architectures

The PTree distribution architecture obtains a performance improvement with respect to the simple tree architecture because it exploits all the P2P transmission resources.

PTree is a particular type of mesh network, characterized by a specific architecture that requires a non marginal effort to build. Extending our analytical model to a generic mesh is possibly not feasible (though we are exploring this direction). On the other hand, the Monte Carlo technique cannot be applied to an unstructured distribution architecture. The reason is that without any structure the dimension of the space to be analyzed explodes due to possible topologies, the multiple bandwidths and the chunk availability at any peer. However, it can be extended to specific mesh architectures with less constraints than the PTree one, and we are working in this direction too.

VII. Related Work

Performance analysis of P2P systems is still in its infancy, and there are only few works on the argument. Most of the analytical works analyzes a specific P2P system, and often the problem is stated very differently from the perspective of this paper. The work in [8] is among the first to evaluate the performance of a P2P system, through the representation as a multi-class closed queuing network. In [9], authors use an age dependent branching process to model the transient evolution of a P2P system and a simple Markovian model to analyze the steady state regime. Fluid models have been recently considered since they can efficiently describe such an amount of transferred data. The work in [10] proposes a fluid model for the analysis of the Squirrel protocol [11]. The result is an accurate model that estimates the performance of the protocol. In [12] authors study the BitTorrent protocol with a simple fluid model. The model is able to catch the transient and the steady state behavior of the system with few simple parameters; moreover, an analysis of the different mechanisms of BitTorrent is provided. Of the works above only [8] tackles the problem of presence of different access bandwidths among peers which is instead the one of the purposes of this paper.

A related topic where distribution architectures are explicitly taken into account is the delivery of streaming services through overlay multicast. Narada [13], ALMI [14], NICE [15], and SplitStream [4] (from which the inspiration for PTree was taken), for instance, define a set of mechanisms to efficiently distribute the content to many overlay nodes. They build in different ways distribution trees and manage the dynamics of leaving and joining nodes. Nevertheless most of these studies are focused on protocol design and do not analyze the impact of distribution architectures on performance. Performance evaluation is only focused on the proposed protocol.

Other studies, [16] and [17], analyze file swarming but do not consider any particular architecture and are focused on other problems, like replication strategies and peer selection. The work in [18] studies how to build the tree topology, but it does not compare different topologies.

At the best of our knowledge no comparative studies have been done so far for different distribution architectures, evaluating the benefits of different policies in the distribution process. Only one work [19] defines, as we do, a model for chain based and tree based architectures, but it analyzes the system with max-plus algebra considering an infinite number of packets, calculating the long term average of the throughput; our analysis instead considers a finite file size and calculates the probability density function at each step of the file distribution process.

VIII. Discussion and Conclusions

Collaborative content delivery is one of the possible applications of P2P networking. Indeed, it may relieve flash-crowds phenomena, and it can also be coupled with "push" based services: the owner of a content (e.g., software firms releasing major patches to subscribing customers) distributes the content on the P2P distribution architecture instead of broadcasting a mail for the customers to download the software from the server.

In these scenarios, besides the more traditional (in P2P networking) aspects such as trust management, overlay management, information retrieval, the distribution performance assumes a major importance, specially when time-critical data is involved.

In this paper we discussed the performance of three simple distribution architectures in presence of heterogeneous peer bandwidth. We introduced an analytical model that provides a closed form solution assuming no correlation among successive distribution steps. The model proved to be very accurate due to a weak correlation structure of the distribution process.

The results obtained have been validated against Monte Carlo simulations. Both the analytical model and the Monte Carlo solution yield the pdf of the distribution process, as a function of time and as a function of the number of nodes reached.

The insight given by the model has been used to devise a modified dynamic distribution architectures that enable the delivery to "get around" slow peers even when the knowledge about the peer bandwidth is limited or null. Indeed, the solution of dynamically modifying the degree of the distribution tree is not entirely novel, since it has been used in many applications (mainly, protocols for streaming services, some of them discussed in Sect. VII). To the best of our knowledge, however, this is the first time its impact has been quantified as a function of the upper and lower bounds of the degree and a thorough analysis has been carried out.

Finally the presence of free-riders has been taken into account, showing that the selfish peers have not a great impact on perfromances. Although this might be intuitive, the quantitative analysis gives a robust method to evaluate whether it is necessary to introduce countermeasures in the application or the basic distribution architecture is robust enough to tolerate some misbehavior.

ACKNOWLEDGMENT

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

REFERENCES

[1] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking Up Data in P2P Systems," *Communications of the ACM*, vol. 46 (2), pp. 43–48, Feb. 2003.

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

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

[4] 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.

[5] 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.

[6] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno, "Analysis of Resource Transfer in Peer-to-Peer File Sharing Applications using Fluid Models," To appear in *Performance Evaluation, Special Issue on Performance Modeling and Evaluation of Peer-to-Peer Computing Systems*.

[7] E. Adar, and B. A. Huberman, "Free Riding on Gnutella," in First Monday, Vol. 5, No. 10, October 2000.

[8] 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.

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

[10] F. Clevenot and P. Nain, "A Simple Fluid Model for the Analysis of the Squirrel Peer-to-Peer Caching System," in *Proc. IEEE INFOCOM 2004*, Hong Kong, Mar. 2004.

[11] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: a Decentralized Peer-to-Peer Web Cache," in *Proc. ACM Symposium on Principles of Distributed Computing (PODC 02)*, Monterey, California, 2002.

[12] 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.

[13] Y.-H. Chu, S. G. Rao, and H. Zang, "A Case for End System Multicast," in *Proc. of ACM SIGMETRICS 2000*, June 2000.

[14] 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.

[15] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proc. SIGCOMM 2002*, Aug. 2002.

[16] L. Massoulie and M. Vojnovic, "Coupon Replication Systems," in *Proc. ACM Sigmetrics 2005*, Banff, Alberta, Canada, June 2005.

[17] D. Stutzbach, D. Zappala, and R. Rejaie, "The Scalability of Swarming Peer-to-Peer Content Delivery," in *Proc. of Networking 2005*, Waterloo, Ontario, Canada, May 2005.

[18] S.-W. Tan, A. G. Waters, and J. Crawford, "Meshtree: A Delay optimised Overlay Multicast Tree Building Protocol," Univ. of Kent, Tech. Rep. 5-05, April 2005.

[19] 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.

[20] The Octave Web Page. `http://www.octave.org/`