UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

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

INTRODUCING HETEROGENEITY IN PERFORMANCE ANALYSIS OF P2P NETWORKS FOR FILE DISTRIBUTION.

Damiano Carra

Renato Lo Cigno

Ernst W. Biersack

December 2004

Technical Report # DIT-04-113

Introducing Heterogeneity in Performance Analysis of P2P Networks for File Distribution*

D. Carra^1, R. Lo Cigno^1, E. W. Biersack^2

^{1}Dipartimento di Informatica e Telecomunicazioni – Università di Trento

^{2}Institut EURECOM, Sophia Antipolis, France

January 31, 2005

Keywords: Peer-to-peer, file distribution, performance evaluation

Abstract

Peer-to-Peer networks have been widely used for file sharing and file distribution. This technical report investigates the performances that can be obtained in a system where two classes of users are present. The study introduces a set of scheme that can be used with different distribution architectures and evaluates the performances that can be obtained with each scheme. Results show that, when we consider heterogeneity, performances may not negatively be affected with respect to the case of single class of users.

1 Introduction

Popularity of Peer-to-Peer (P2P) networks is increasing quickly thanks to their characteristics: in P2P system nodes are equal and can act as a client, as a server and as a router. This approach provides important benefits, such as improved scalability and fault-tolerance, and cost maintenance reduction due to self organizing nature of the network.

P2P model has been used for many different applications: distributed storage [1], service directory, grid computing [2]. Among all these applications, file sharing and content distribution is the most widespread one. Most of the solutions proposed so far focus their attention on the content search phase, while only recently efficient download has been taken into account [3].

In this technical report we start from the analysis of the file distribution architectures made in [4] and consider different scenarios. The main hypothesis that we remove with respect to the previous analysis is the assumption of homogeneous sources: we suppose that peers in the network can have different available bandwidths.

*This work was done under the E-NEXT umbrella

In particular we consider the presence of two classes of peers: for each proposed architecture (Linear, Tree and PTree) we present different schemes that can be applied for the distribution of files. Each class is structured in the above mentioned architectures and the only degree of freedom is how a class can help the other class in order to maximize the overall performance.

We focus our attention on three performance indexes: the total download time necessary to reach N peers, the mean download time and the amount of work done (how many file copies each peer uploads). The results we obtain show that, when we consider heterogeneity, slow peers may not negatively affect the system. In general, the simpler the adopted architecture is, the more improvement can be obtained with a intelligent distribution policy; the more complex the architecture is, the less the gain we have.

The rest of the report is organized as follows. Section 2 analyzes the different architectures and schemes with two classes. Section 3 presents some analytical results obtained with typical values for the number of peers and capacities. Section 4 summarizes the conclusions on the two classes. Finally 5 presents some possible future work related to the subject.

2 Two Classes

This section applies the analysis of the three architectures (Linear, Tree and PTree) proposed in [4] when two classes of users are present in the network. Each class is symmetric, i.e. peers that belong to a class have the upload capacity equal to the download capacity. In case of asymmetric capacities[1], our analysis could give some insights if we consider as reference bandwidth the upload capacity (that is typically the smallest). Nevertheless, we leave for future works the detail comparison of the different scenarios.

Our main metrics of interest are (i) the total download time necessary to N users to complete the download of a file, (ii) the average download time and (iii) the amount of work done by each peer, i.e., how much they upload (this metric can be considered as a fairness index).

The main hypotheses of our model are the following:

- there are only two classes of peers in the network: class 1 (fast peers) with upload and download bandwidth b_1 and class 2 (slow peers) with upload and download bandwidth b_2; $b_1 > b_2$;
- the number of peers in class 1 and class 2 is equal to N_1 and N_2 respectively; the total number of peers is $N = N_1 + N_2$;
- all the peers know all the other peers, including their bandwidths;
- there is exactly one server that has the original content; it is always on-line and it can indefinitely upload the file to new peers;

[1] for example, ADSL users have the download capacity greater than the upload capacity.

- we focus on the distribution of a single file F; the file is divided in C identical pieces called "chunks" and each chunk can be distributed independently;
- the server has sufficient bandwidth to upload concurrently to the two classes, i.e. its upload bandwidth b_S is equal to $b_1 + b_2$; this hypothesis simplifies calculations and does not greatly influence the final results since the impact is only on peers that download from the server[2];
- peers can be selfish, i.e. they disconnect as soon as they finish downloading the content, or altruistic, i.e. they remain in the system for a certain period of time (that is related to the specific used policy).

Even if the hypothesis of global knowledge can be unrealistic, it is necessary in order to understand the optimal case. As future work we will remove this hypothesis analyzing the effect of limited knowledge. On the contrary, the hypothesis of knowing to some extent the capacity of each link is realistic, since most of the current systems can estimate the bandwidth of a peer during previous exchanges (and each peer can also communicate its own capacity). Here we assume perfect knowledge; an error function can be added in future works.

Aim of this study is to find optimal policies for file distribution in such environment; depending on the file distribution policy, the performance can be greatly different.

2.1 Linear

In this particular architecture, parallel upload should not be considered; nevertheless, in order to exploit the capability of the system it is necessary to allow some special configuration with parallel upload. For instance, it is possible for a fast peer to upload in parallel to exactly one slow peer, with bandwidth b_2, and one fast peer, with bandwidth $(b_1 - b_2)$. Another case is when fast peers are altruistic, so they use their capacities to help slow peers.

Under the hypotheses made above it is possible to identify a set of cases. In the following paragraphs we will analyze the cases considered most interesting.

Figure 1 shows a basic scheme to calculate the number of peers in time. In the left part it is shown the evolution when there is a single class: each row corresponds to a single chain. The first peer finishes to download after F/b_1 rounds; the second peer finishes at $F/b_1 + F/Cb_1$ since it has to wait that first peer has downloaded the first chunk before starting the upload to the second peer. A new chain is started when the file is completely download from the first peer. In the right part we consider the case with two classes when a server alternatively serves a fast peer (with bandwidth b_1) and a slow peer (with bandwidth b_2).

[2] We are interested in keeping the number of peers that download from the server as small as possible, because the work must be shared among peers.

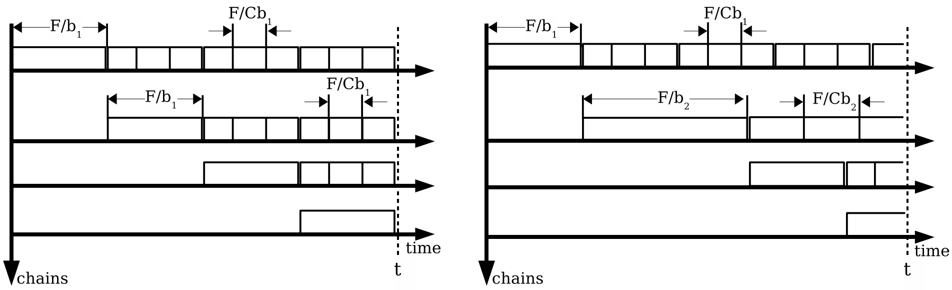


Figure 1: Linear chain: number of peers per chain versus time (C=3) with Linear architecture: the server upload to a single class (left) and alternatively to two classes (right)

If we observe the system at certain time t it is possible to calculate the number of chains and the number of peers. In general, through this kind of scheme we can evaluate the number of peers versus time for all the policies we have defined.

2.1.1 Independent

In this case the server uploads chunks independently to each class; peers belonging to a class do not exchange contents with other class's peers. Figure 2 shows the chunk distribution methodology.

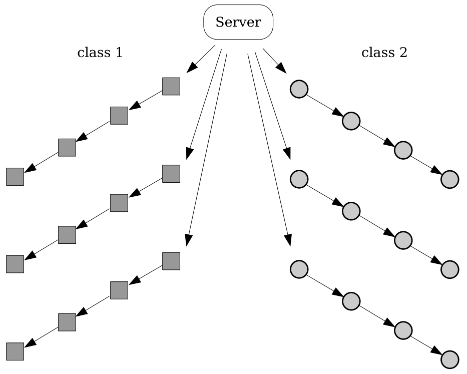


Figure 2: Chunk distribution with two independent classes

Total download time. Generalizing the method used in [4], it is simple to find that the time necessary to distribute the content to N_i peers with bandwidth b_i with a Linear scheme is

$$T_{\text{Linear}}(b_i, C, N_i) = \frac{F}{b_i} \cdot \frac{(C-2) + \sqrt{(C-2)^2 + 8N_iC}}{2C} \tag{1}$$

where F is the file size in bits and b_i the capacity of the class i in bit/s. Since the two classes evolve independently, the total time necessary to reach N peers is dominated by slow peers. Figure 3 shows the total time against the number of peer n: in this example we have two classes with the same number of peers, i.e. $N_1 = N_2 = N/2$, where $N = 10^4$, and different bandwidth ratios: we assume that the bandwidth of class 2 is fixed, with $F/b_2 = 1$ and the bandwidth of class 1 is two, five, ten and 100 times greater.

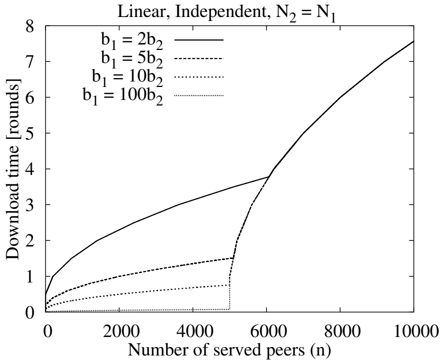


Figure 3: Linear chain with independent classes: time necessary to complete the download

Average download time. Another important metric that can be useful to understand the performance is the average download time. We consider the sum of the times necessary to complete the download of each peer and then we divide it by the number of peers.

Referring to Fig. 1 (left part), consider the first chain. The first peer finishes to download at time $\frac{F}{b_i}$, the second peer at time $\frac{F}{b_i} + \frac{F}{Cb_i}$, the third at time $\frac{F}{b_i} + 2\frac{F}{Cb_i}$ and so forth. So at time t in the first chain the sum of each single download time is

$$\sum_{j=0}^{\frac{t-F/b_i}{F/Cb_i}} \frac{F}{b_i} + j\frac{F}{Cb_i} \,. \tag{2}$$

The second chain is equal to the first, with a delay of $\frac{F}{b_i}$, and the third chain is delayed $2\frac{F}{b_i}$. Let $t_{\text{class i}}$ the time necessary to complete all the downloads of a specific class i, i.e., $t_{\text{class i}} = T_{\text{Linear}}(b_i, C, N_i)$ (we will use the simplified form $t_{\text{class i}}$ where it is clear which architecture and scheme we are using, otherwise we will use the complete form). Since in the system there are $\frac{t_{\text{class i}}}{F/b_i}$ chains, the sum of the total download times D_i for each class is

$$D_i = \sum_{k=1}^{\frac{t_{\text{class i}}}{F/b_i}} \sum_{j=0}^{\frac{t_{\text{class i}}-kF/b_i}{F/Cb_i}} k\frac{F}{b_i} + j\frac{F}{Cb_i} \,. \tag{3}$$

The value of $t_{\text{class i}}$ can be found with (1). The average download time is D_i/N_i for each class and $(D_1 + D_2)/(N_1 + N_2)$ globally.

Amount of work. The number of copies distributed by the server is equal to the number of started chains for each class. In this case we have $\sum_i \frac{t_{\text{class i}}}{F/b_i}$.

As regards fast and slow peers, each peer uploads once. Only the last peer of each chain does not upload; we can consider that the impact on the index is not significant.

2.1.2 Generous

With this configuration slow peers do not upload any chunk; they only download from fast peers; fast peers upload in parallel the chunks to a fast peer and to a slow peer. Each fast peer stops after it has completely served one slow peer. Figure 4 shows the chunk distribution methodology in this case.

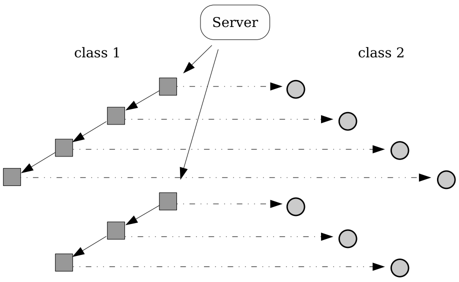
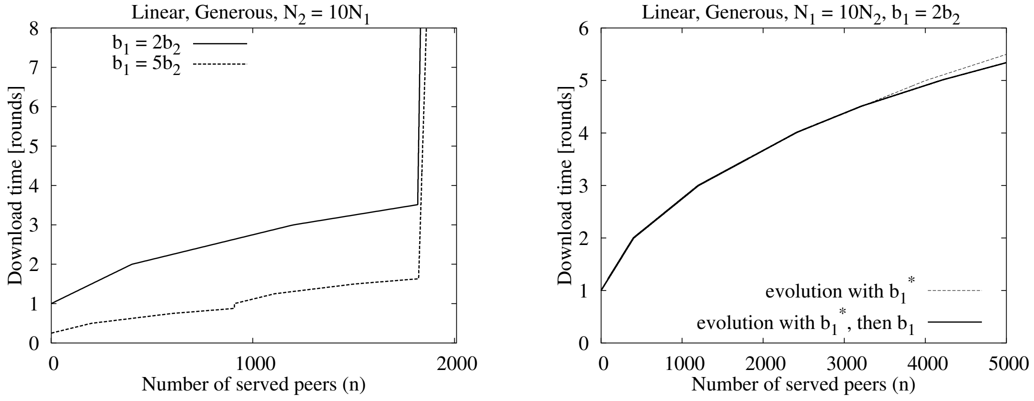


Figure 4: Chunk distribution with generous fast peers

Total download time. The system at the beginning evolves as if there were only a single class with capacity $b_1^* = b_1 - b_2$. Every helped slow peer finishes to download $\frac{F}{b_2} + \frac{F}{Cb_1^*} - \frac{F}{b_1^*}$ rounds after the correspondent fast peer terminates. When the peers of one class finish downloading, the evolution of the system is different depending whether N_1 is larger or smaller than N_2. Let n be the number of peers that have already finished downloading at time t, $0 < n < N$. If $N_1 < N_2$ then fast peers finish before slow ones (see Fig. 4) and, when $n > 2N_1$, the remaining slow peers can only download form the server (they are not collaborative, so they don't upload to any other peers); in this case b_S/b_2 peers finish to download every F/b_2. If $N_1 > N_2$ then slow peers finish before fast ones and, when $n > 2N_2$, the remaining fast peers evolves with full bandwidth b_1.

Figure 5 shows the behavior in two cases. When $N_1 < N_2$ (here $N_2 = 10N_1$, with $N_1 \simeq 900$) it is possible to see that, after $2N_1$, the system evolves very slowly, since only the server uploads the content. On the contrary, when $N_1 > N_2$ (here $N_1 = 10N_2$, with $N_2 \simeq 900$), only a small part of fast peer are involved in helping slow peers; after $2N_2$ peers are served, the system evolves faster. The Figure shows how the system evolves: to see the difference between the phase when class 1 has a capacity equal to b_1^* and when it has a capacity equal to b_1, the dashed line represents the evolution if class had always a capacity b_1^*. In case of greater bandwith ratios (not shown here), the difference becomes not significant.



(a) $N_1 < N_2$

(b) $N_1 > N_2$

Figure 5: Linear chain with generous fast peer: time necessary to complete the download. Left side ($N_1 < N_2$): when fast peers have completed, slow peers can download only from the server. Right side ($N_1 > N_2$): after helping slow peers, fast peers evolve with full bandwidth

Average download time. The average download time has to be calculated in the two different cases.

When $N_1 < N_2$, the average download time of the N_1 fast peers can be calculated with (3), where the bandwidth is $b_1^* = b_1 - b_2$. For class 2, each of the N_1 slow peers finishes with a delay of $\frac{F}{b_2} + \frac{F}{Cb_1^*} - \frac{F}{b_1^*}$ with respect each fast peer, so we can use again (3), with $b_i = b_1^*$ and considering that delay by simply adding at the end $N_1\left(\frac{F}{b_2} + \frac{F}{Cb_1^*} - \frac{F}{b_1^*}\right)$. The remaining $N_2 - N_1$ slow peers complete at the rate of b_S/b_2 peers every F/b_2. So the sum of all the delays experimented by these peers is

$$\sum_{k=1}^{\frac{N-2N_1}{b_S/b_2}} t_{\text{class1}} + k\frac{F}{b_2} \tag{4}$$

where t_{class1} is the time necessary to class 1 to complete.

When $N_1 > N_2$ we can approximate the average download time of class 1 with (3) using the bandwidth b_1^*. For class 2, we can calculate the sum of the delays with (3), with bandwidth b_1^*; the value of $t_{\text{class i}}$ can be calculated from (1) using a number of peers equal to N_2; finally the term $N_2\left(\frac{F}{b_2} + \frac{F}{Cb_1^*} - \frac{F}{b_1^*}\right)$ must be added in order to consider the delay necessary to complete the download.

Amount of work. The number of copies distributed by the server is equal to the number of started fast chains. If the number of fast peer is not sufficient to serve all the slow peers, the server must upload to the remaining slow peers, so we have $\frac{t_{\text{class i}}}{F/b_i^*} + \max(0, (N_2 - N_1))$.

This represent an approximation since we do not distinguish between $N_1 > N_2$ and $N_1 < N_2$; nevertheless this approximation has not a great impact on final results.

As regards peers, each fast peer uploads at most twice (depending on the number of slow peers), whereas slow peers do not upload.

2.1.3 Generous, with Collaboration

In this configuration the system evolves as in the previous case, but here each slow peer served by a fast peer starts a new chain. In this case each fast peer serves one fast peer and one slow peer, and each slow peer serve one slow peer. Figure 6 shows the chunk distribution methodology in this case.

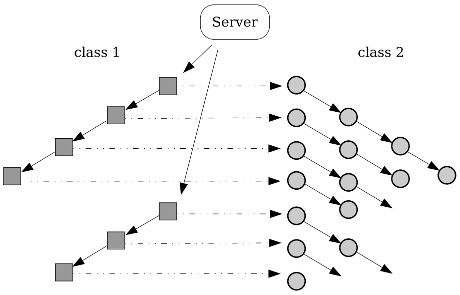


Figure 6: Chunk distribution with generous fast peers and collaborative slow peers

With this scheme we try to exploit the unused capacity of slow peers. While fast peers continue to upload chunks to slow peers, each slow peer starts a new chain. The rate of slow chain creation is equal to the rate new fast peers are involved in the distribution process.

Total download time. Looking at fast peers, the download time can be found, as in the previous case, simply considering a Linear evolution with capacity $b_1^* = b_1 - b_2$. In case of class 2 complete before class 1, the remaining fast peers evolve with full bandwidth b_1.

As concern slow peers, in order to find the total download time we first find the number of slow peers that have completed the download at time t and then it is possible to derive the formula of total download time against the number of served slow peers.

Consider the first chain of fast peers. A new fast peer is reached by a chunk every F/Cb_1^*, so the number of fast peer in the first fast chain is $\frac{t_{\text{class1}} - F/b_1^*}{F/Cb_1^*} + 1$. For each fast peer a new slow chain is started. The number of slow peers in a slow chain at time t is $1 + \frac{t - t_{\text{start}} - F/b_2}{F/Cb_2}$, where t_{start} is the time when the chain is started. The number of slow peers contained in all the slow chains generated by the first fast chain is then

$$\sum_{j=1}^{\frac{t_{\text{class1}} - F/b_1^*}{F/Cb_1^*}+1} \max\left(0, \left\lfloor 1 + \frac{t - j\frac{F}{Cb_1^*} - \frac{F}{b_2}}{F/Cb_2} \right\rfloor\right) \tag{5}$$

where t_{class1} is the time necessary to class 1 to complete.

The second fast chain generates a number of slow chains equal to $\frac{t_{\text{class1}}-2F/b_1^*}{F/Cb_1^*}$, so the number of slow peers contained in all the slow chains generated by the second fast chain is

$$\sum_{j=1}^{\frac{t_{\text{class1}}-2F/b_1^*}{F/Cb_1^*}+1} \max\left(0, \left\lfloor 1+\frac{t-\left(j\frac{F}{Cb_1^*}+\frac{F}{b_1^*}\right)-\frac{F}{b_2}}{F/Cb_2}\right\rfloor\right) . \tag{6}$$

Applying the calculus for every fast peer chain, we obtain the total number of slow peer reached at time t

$$n_2(t)=\sum_{k=0}^{\frac{t_{\text{class1}}}{F/b_1^*}} \sum_{j=1}^{\frac{t_{\text{class1}}-(k+1)F/b_1^*}{F/Cb_1^*}+1} max\left(0, \left\lfloor 1+\frac{t-\left(j\frac{F}{Cb_1^*}+k\frac{F}{b_1^*}\right)-\frac{F}{b_2}}{F/Cb_2}\right\rfloor\right) . \tag{7}$$

From this relation, it is possible to find t_{class2}, time necessary to class 2 to complete, such that $n_2(t_{\text{class2}}) = N_2$. It is important to note that equation (7) does not take into account the possible contribution of the server, if it becomes available (this happens if class 1 terminates before all slow peers finish): nevertheless we consider the contribution not significant, since there are a lot of slow chain started (a slow chain for every fast peer) that generates a lot of slow peers every F/Cb_2.

Average download time. As regards the average download time, we have to find the sum of the delay experimented by the N peers. For class 1 the delays can be found with the approximation that in both cases, $t_{\text{class1}} < t_{\text{class2}}$ and $t_{\text{class1}} > t_{\text{class2}}$, the bandwidth is b_1^* and then apply (3).

For class 2, consider the slow chains born from the first fast chain: the first peer of the first slow chain finishes at $\frac{F}{Cb_1^*}+\frac{F}{b_2}$, the second at $\frac{F}{Cb_1^*}+\frac{F}{b_2}+\frac{F}{Cb_2}$, the third at $\frac{F}{Cb_1^*}+\frac{F}{b_2}+2\frac{F}{Cb_2}$; the first peer of the second slow chain finishes at $2\frac{F}{Cb_1^*}+\frac{F}{b_2}$, the second at $2\frac{F}{Cb_1^*}+\frac{F}{b_2}+\frac{F}{Cb_2}$, the third at $2\frac{F}{Cb_1^*}+\frac{F}{b_2}+2\frac{F}{Cb_2}$; generalizing for all the $\frac{t_{\text{class1}}-F/b_1^*}{F/Cb_1^*}+1$ slow chains derived from the first fast chain we obtain:

$$\sum_{k=1}^{\frac{t_{\text{class1}}-F/b_1^*}{F/Cb_1^*}+1} \sum_{j=0}^{\frac{t-k\frac{F}{Cb_1^*}-\frac{F}{b_2}}{F/Cb_2}} \left(k\frac{F}{Cb_1^*}+\frac{F}{b_2}+j\frac{F}{Cb_2}\right) \tag{8}$$

where t is the instant that we choose to observe the system. If we consider the other fast peer chains we obtain:

$$D_2=\sum_{l=0}^{\frac{t_{\text{class1}}}{F/b_1^*}} \sum_{k=1}^{\frac{t_{\text{class1}}-(l+1)\frac{F}{b_1^*}}{F/Cb_1^*}+1} \sum_{j=0}^{\frac{t-l\frac{F}{b_1^*}-k\frac{F}{Cb_1^*}-\frac{F}{b_2}}{F/Cb_2}} \left(l\frac{F}{b_1^*}+k\frac{F}{Cb_1^*}+\frac{F}{b_2}+j\frac{F}{Cb_2}\right) . \tag{9}$$

The result of this formula is cumbersome and can be evaluated via software implementation.

Amount of work. The number of copies distributed by the server is equal to the number of started fast chains. Since slow peers uploads, even if fast peers complete before slow peers, it is not necessary that the server continue to upload to slow peers.

As regards peers, each fast peer uploads at most twice, whereas each slow peer uploads at most once (last peer of each chain does not upload; since here the number of chains is high, this term becomes significant).

2.1.4 Altruistic

In this configuration each fast peer, after uploading the content to a fast peer, stays on-line and serves slow peers; in this case, since the capacity is big, parallel upload is necessary to exploit all the fast peers upload capacity. This means that b_1/b_2 slow peers start to download from a single fast peer and finish to download after F/b_2 rounds. For simplicity, we suppose no collaboration of slow peers, i.e. they do not start new chains. We suppose that fast peers are able to serve all the slow peers (i.e. $\frac{b_1}{b_2}N_1 > N_2$), so they upload only once. Figure 7 shows the chunk distribution methodology in this case.

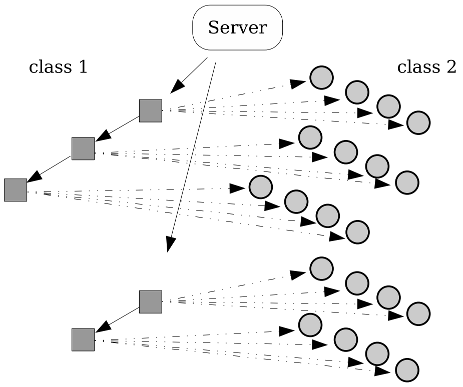


Figure 7: Chunk distribution with altruistic fast peers

Total download time. Considering an instant t, the number of slow peers that has completed is equal to the number of fast peer that have completed at time $t - F/b_2$ multiplied by a factor b_1/b_2. The total time for class 2 to complete is $t_{\text{class2}} = F/b_2 + T_{\text{Linear}}(b_1, C, \frac{N_2}{b_1/b_2})$, where the last term is the time, found with (1), necessary to reach a number of fast peers that is able to serve all the slow peers (since every fast peer serves $\frac{b_1}{b_2}$ slow peers, $\frac{N_2}{b_1/b_2}$ fast peer are needed).

Average download time. For class 1 we can use (3). For class 2, following the steps made to find (3), considering that each fast peer uploads the file to b_1/b_2 slow peers and they finish with a delay of F/b_2 with respect to the fast peer, the mean download time is

$$D_2 = \sum_{k=1}^{\frac{t_{\text{class2}} - F/b_2}{F/b_1}} \sum_{j=0}^{\frac{t_{\text{class2}} - F/b_2 - kF/b_1}{F/Cb_1}} \frac{b_1}{b_2}\left(k\frac{F}{b_1} + j\frac{F}{Cb_1} + \frac{F}{b_2}\right) . \qquad (10)$$

The Altruistic case is the most interesting case because it removes the hypothesis that peers leave the system as soon as they complete, so they can help slow peers. Of course this comes at a cost: each fast peer replicates the file b_1/b_2 times. In a specific context, this could be not acceptable (private users); in other context (file distribution in a company) this is not a problem.

Amount of work. The number of copies distributed by the server is equal to the number of started fast chains. As regards peers, each fast peer uploads $1 + \frac{N_2}{N_1}$ times, whereas slow peers do not upload.

2.2 Tree

With Tree architecture it is possible to have parallel uploads. In the general case, it is possible to specify different outdegrees: k, the outdegree of a fast peer toward fast peers; f, the outdegree of a fast peer toward slow peers; s the outdegree of a slow peer toward slow peers. Thanks to these design parameters, it is possible to organize the peers such that they all finish the download at the same time: this optimizes the total download time that is our main performance index.

As in the Linear architecture, it is possible to identify different cases. In the following paragraphs we analyze them.

2.2.1 Independent

Total download time. Each class builds its own tree and evolves independently; no chunks are exchanged between the two trees. The total download time is the time necessary to the first chunk to reach the leaves plus the time to upload the file to the leaves using a bandwidth b_i/k for each upload[3]. The number of levels l in a tree with N_i nodes can be found considering that the first level contains k nodes (the level 0 is the server itself), the second $k \cdot k$ nodes and so forth; so $N_i = \sum_{j=1}^{l} k^j = k\frac{k^l-1}{k-1}$ and $l = log_k\left(N_i\frac{k-1}{k} + 1\right)$. If the tree is not full, we can consider $l = \left\lceil log_k\left(N_i\frac{k-1}{k}\right)\right\rceil$ (we also suppose that $N_i\frac{k-1}{k} \gg 1$). A single chunk reaches the leaves at time $\frac{F/C}{b_i/k}l = \frac{F/C}{b_i/k}\left\lceil log_k\left(N_i\frac{k-1}{k}\right)\right\rceil$, so the total download time is

$$T_{\text{Tree}}(b_i, C, k, N_i) = \frac{F}{b_i}k + \frac{F}{b_i}\frac{k}{C}\left(\left\lceil log_k\left(N_i\frac{k-1}{k}\right)\right\rceil - 1\right) \tag{11}$$

where, we subtract 1 to the number of levels because the time to upload the first chunk to the leaves is included in the time to upload the whole file. If the class 2 is considered, the parameter k has to be substituted by the parameter s. Since there are no chunk exchanges between trees, the parameter f is zero.

For the optimization problem of the outdegrees k and s, with this scheme it is not correct to adjust the parameters in order to obtain the same download time for the two

[3] For simplicity, we use the notation with k for both fast and slow trees.

classes: in fact, since they are independent, the modification of download time of one class does not influence the performance of the other class. This means that in this case we have to optimize the parameters independently, with the methodology proposed in [4]; then the optimal values are $k = s = 2$.

Figure 8 shows the total time against the number of peer n with two classes with the same number of peers ($N_1 = N_2 = N/2$, where $N = 10^4$), and different bandwidth ratios: we assume that the bandwidth of class 2 is fixed, with $F/b_2 = 1$ and the bandwidth of class 1 is two, five, ten and 100 times greater.

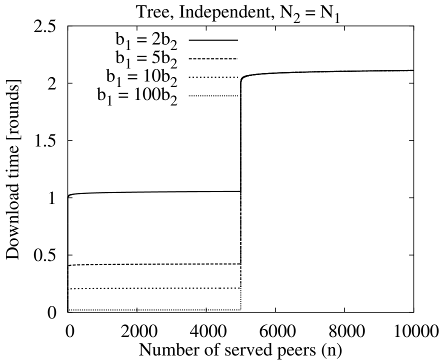


Figure 8: Tree architecture with independent classes: time necessary to complete the download

Average download time. We consider the sum of the times necessary to complete the download of each peer and then we divide it by the number of peers. At time $\frac{F}{b_i/k}$ the first k peers complete; at time $\frac{F}{b_i/k} + \frac{F}{Cb_i/k}$ other $k \cdot k$ peers complete; at time $\frac{F}{b_i/k} + 2\frac{F}{Cb_i/k}$ other $k \cdot k \cdot k$ peers complete. Generalizing we have

$$D_i = \sum_{j=0}^{\lceil log_k(N_i(k-1)/k) \rceil} k^{j+1} \left(\frac{F}{b_i/k} + j \frac{F}{Cb_i/k} \right) . \tag{12}$$

The above formula is valid for class 1; for class 2 instead of k the outdegree s has to be considered. The average download time is D_i/N_i for each class and $(D_1+D_2)/(N_1+N_2)$ globally.

Amount of work. The number of copies distributed by the server is equal to the number of started roots for each class, i.e., $k + s$. As regards peers, each interior node uploads k (fast peers) or s (slow peers) times, whereas each leaf node does not upload.

2.2.2 Generous

Slow peers do not upload any chunk, they only download from fast peers. Fast peers evolve according to a Tree architecture with outdegree k and capacity $b_1^* = b_1 - f \cdot b_2$; at the same time, they upload the chunks in parallel to f slow peers with bandwidth b_2 each. This means that, once the leaves of the tree formed by fast peers are reached, the time necessary to fast peers to complete is $k\frac{F}{b_1^*}$ rounds, while the time needed for slow peers to finish is equal to $\frac{F}{b_2}$, without the factor k that slows down the performance.

For the optimization problem, we have to find the best value for k and f (in this case $s = 0$). The optimum value k for the tree built by fast peer is independent from the fact that part of the bandwidth is dedicated to slow peers, so $k = 2$. The optimum value f must be chosen considering that we minimize the total download time if all the peers finish as much as possible together: to do so, all the N_1 generous fast peer must upload to all the N_2 slow peers, so $f = \left\lceil \frac{N_2}{N_1} \right\rceil$. Nevertheless, if f is sufficiently big, the bandwidth for class 1 ($b_1^* = b_1 - f \cdot b_2$) could be smaller than b_2 (or, in the worst case, less than zero) and the total download time would be greater than in the Independent case: this result is due to the non-collaborative behavior of class 2. We can conclude that $b_1 - f \cdot b_2$ should be greater than b_2, i.e. $f < \frac{b_1}{b_2} - 1$; this means that, if $b_1 < (\frac{N_2}{N_1} + 1)b_2$ the scheme is not applicable.

Total download time. Class 1 evolves with bandwidth $b_1^* = b_1 - f \cdot b_2$. The total time for this class can be found simply by substituting the value of the bandwidth in (11). As regards class 2, the download completes after F/b_2 rounds the first chunk of class 1 has been completely uploaded to the leaves of the fast tree, so

$$T_{\text{Tree}}^{\text{Class2}}(b_2, C, k, N_2) = \frac{F}{b_1^*}\frac{k}{C}\left\lceil log_k \left(N_1 \frac{k-1}{k} \right) \right\rceil + \frac{F}{b_2} \, . \tag{13}$$

It is important to note that in a Tree architecture leaf nodes of fast tree do not upload to other fast peers, but only to slow peers. The unused capacity of leaf nodes (b_1^*) can be exploit using an *alternative* Generous scheme. We can impose that interior nodes of fast tree upload only to other k fast nodes, whereas leaf nodes, as soon as they receive the first chunk, upload to all the slow peers. The number of leaves in a tree is $k^{l-1} < \#leaves < k^l$ (the tree could be not full), i.e. $\frac{k-1}{k^2}N_i < \#leaves < \frac{k-1}{k}N_i$. In the worst case, i.e., when only a node stays at lowest level, the number of leaves is approximately $\frac{k-1}{k^2}N_i$ and each leaf node has to upload up to $\frac{k^2}{k-1}\frac{N_2}{N_1}$ slow peers with bandwidth b_2. This means that this alternative scheme is not applicable if $b_1 < \frac{k^2}{k-1}\frac{N_2}{N_1}b_2$. With respect the previous Generous scheme, this constraint imposes higher capacity b_1 when the ratio between the number of peers of each class grows. If the constraint is satisfied, the total download time for class 1 is equal to the total download time in the Independent case, since interior nodes can use all the capacity. For class 2, it is sufficient to substitute b_1 to b_1^* in (13).

Average download time. The metric can be found with (12). For class 1 we substitute $b_i = b_1^*$, finding D_1. For class 2, the average is equal to the average of class 1 added with a term $\frac{F}{b_2} + \frac{F}{b_1^*}\frac{k}{C} - \frac{F}{b_1^*}k$ that is the delay that each slow peer experiments with respect to a peer of class 1.

In the case of the alternative Generous scheme, for class 1 b_1 instead of b_1^* must be used in (12). For class 2, the average is simple the total download time, since all the slow peers start and finish together [4].

Amount of work. The number of copies distributed by the server is equal to the number of fast roots, i.e., k. As regards peers, each fast peer uploads at most $k+\frac{N_2}{N_1}$ times, whereas slow peers do not upload. In particular, interior nodes of fast peer upload in parallel to k fast peers and to $\frac{N_2}{N_1}$ slow peers and leaf nodes upload to $\frac{N_2}{N_1}$ slow peers.

In case of the alternative Generous scheme, interior nodes upload k times, whereas leaf nodes upload up to $\frac{k^2}{k-1}\frac{N_2}{N_1}$ times.

2.2.3 Generous, with Collaboration

Total download time. With this scheme, we try to exploit the unused capacity of the slow peers already served: once a slow peer has received a chunk, it starts to redistribute it. The optimal outdegree s of slow peer trees is 1: in fact, from (11), the time necessary to complete a tree is at least equal to $\frac{F}{b_2}s$ and a value of s greater than 1 implies that collaboration would not have effect, as it becomes evident through the comparison between (11), with $s > 1$, and (13)[5]. Figure 9 shows an example of the scheme.

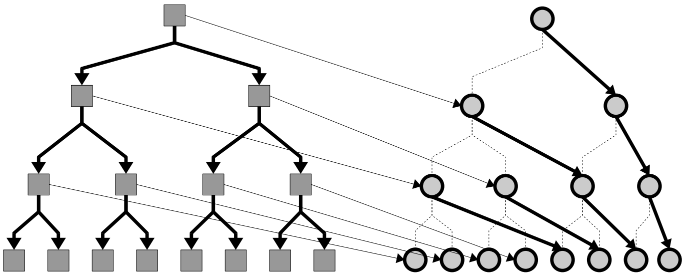


Figure 9: Tree architecture: tree building in case of k=2 and s=1

It is important to note that introducing collaboration influences class 1 performance, but has only a little impact on class 2 results. In fact collaborative behavior diminishes the parameter f and the capacity b_1^* becomes greater; nevertheless the capacity b_1^* is present only in the logarithmic term of (13) that gives the smallest contribution to the total time.

In order to reduce the number of parameters, we can choose k considering that fast trees evolve independently (so $k = 2$), having only the rate of creation of slow chains (the parameter f) to find. We start from the maximum value of f, i.e., $f = N_2/N_1$ and we calculate the total download time in this case (that is the Generous case, without collaboration, since fast peers help all the slow peers); we then iteratively reduce the value

[4] Even if leaf nodes in the fast tree can stay in different levels, the difference in terms of delay can be considered not significant.

[5] If $b_1 \simeq b_2$ and $log_k(N_1) \gg log_s(N_2)$ this could not be true, but these degenerate cases are not considered.

of f calculating if, with the collaboration of slow peers, it is possible to reach all the slow peers in lower time (with respect to the previous iteration).

Once the value f is found, class 1 total download time is given by (11) with $b_i = b_1^* = b_1 - fb_2$. Even if slow peers collaborate in distributing the content, all the fast peers upload the content to a subset of slow peers (each fast peer uploads to f slow peers), so the bandwidth b_1^* does not change during the download. As concerns class 2, we have to analyze the behavior in detail. At the beginning, the first k fast peers finish to upload the first chunk to the $k \cdot f$ slow peers at time $\frac{kF}{Cb_1^*} + \frac{F}{Cb_2}$ (the first term is the time necessary to download the first chunk and the second term is the time necessary to upload it with bandwidth b_2). Therefore the $k \cdot f$ slow peers become heads of Linear chains that distribute the content with bandwidth b_2. The number of peers in these chains at time t is $kf(1 + \frac{t-(kF/Cb_1^*+F/b_2)}{F/Cb_2})$. In the meanwhile the second level of the fast tree uploads the content to new slow peers: we obtain $k \cdot k \cdot f$ new slow peers. The number of slow peers in these chains is $kkf(1 + \frac{t-(2kF/Cb_1^*+F/b_2)}{F/Cb_2})$. Generalizing we obtain the number of slow peers ($n_2(t)$) in time

$$n_2(t) = \sum_{j=1}^{\lceil log_k(N_i(k-1)/k) \rceil} k^j f \left(1 + \frac{t - (jkF/Cb_1^* + F/b_2)}{F/Cb_2}\right) . \tag{14}$$

From this relation, it is possible to find t_{class2}, time necessary to class 2 to complete, such that $n_2(t_{\text{class2}}) = N_2$.

As regards the alternative Generous scheme, where interior node of fast tree upload only to fast peers, and leaf nodes upload to slow peers, the collaboration has no impact on the results, since all the slow peers finish together (this is true if the constraint $b_1 > \frac{k^2}{k-1}\frac{N_2}{N_1}b_2$ is satisfied). This alternative scheme could be interesting if the constraint is not satisfied: in this case, leaf nodes upload to as much as possible slow peers and each served slow peer starts a chain as soon as it receives the first chunk. Each leaf node serves b_1/b_2 slow peers, so the total number of chains is (at least) $\frac{k-1}{k^2}N_1\frac{b_1}{b_2}$ and the content is then rapidly distribute to all the slow peers.

Average download time. As regards the average download time, for class 1 we can use (12) with $b_i = b_1 - fb_2$. For class 2, we can apply the procedure used to find (3), finding

$$D_2 = \sum_{l=1}^{\lceil log_k(N_i(k-1)/k) \rceil} k^l f \sum_{j=0}^{\frac{t_{\text{class2}} - (lkF/Cb_1^* + F/b_2)}{F/Cb_2}} \left(l\frac{kF}{Cb_i^*} + \frac{F}{b_2} + j\frac{F}{Cb_2}\right) \tag{15}$$

where t_{class2} is the time necessary to class 2 to complete.

As regards the alternative Generous scheme (in the case the constraint is not satisfied), the average download time for class 2 is equal to the average download time in case of Linear architecture, with a delay equal to the time necessary to start the chains (i.e., to reach the leaves of the fast tree).

Amount of work. The number of copies distributed by the server is equal to the number of fast roots, i.e., k. As regards peers, each fast peer uploads at most $k+\frac{N_2}{N_1}$ times, whereas each slow peer uploads at most once. The alternative Generous scheme (where only leaf nodes are generous) gives the same results as in the previous case (Generous, without collaboration), since slow peers finish all together and can not collaborate to distribute the content.

2.2.4 Altruistic

As in the Linear case, we remove the hypothesis that peers leave the system as soon as they complete. Each fast peer, after uploading the content to k fast peers, stays on-line and starts to serve slow peers; we consider that the number of slow peers is smaller than $\frac{b_1}{b_2}N_1$ (so they upload only once) and slow peers do not collaborate (these hypotheses simplify the calculation, without loss of generality).

Total download time. It is straightforward to see that the time necessary to class 2 to complete is

$$T_{\text{Tree}}^{\text{Class2}}(b_2, C, k, N_2) = \frac{F}{b_2} + T_{\text{Tree}}^{\text{Class1}}(b_1, C, k, \frac{N_2}{b_1/b_2}) \qquad (16)$$

where the last term is the time necessary to reach a number of fast peers that is able to serve all the slow peers (since every fast peer serves $\frac{b_1}{b_2}$ slow peers, $\frac{N_2}{b_1/b_2}$ fast peer are needed).

Average download time. The metric can be found with (12). For class 1 we substitute $b_i = b_1$, finding D_1. For class 2 the average is equal to the average of a fast tree with $\frac{N_2}{b_1/b_2}$ nodes plus $\frac{F}{b_2}$.

Amount of work. The number of copies distributed by the server is equal to the number of fast roots, i.e., k. As regards peers, each fast peer uploads at most $k+\frac{N_2}{N_1}$ times, whereas slow peers do not upload. In this case leaf nodes may not work, depending on the number of slow peers and on the bandwidth ratio. In fact, if the number of interior nodes is sufficient and have sufficient bandwidth to serve all the slow peers, leaf nodes may not upload.

2.3 PTree

As in the Tree architecture it is possible to specify different outdegrees: k, the outdegree of a fast peer toward fast peers; f, the outdegree of a fast peer toward slow peers; s the outdegree of a slow peer toward slow peers.

In the following paragraphs we analyze the different schemes.

2.3.1 Independent

Total download time. Each class builds its own tree and evolves independently; no chunks are exchanged between the two trees. The server starts k different distribution trees and each tree contains N_i nodes.The number of levels l in a tree with N_i nodes can be found considering that the first level contains 1 node, the second k nodes, the third $k \cdot k$ nodes, and so forth; so $N_i = \sum_{j=0}^{l} k^j = \frac{k^{l+1}-1}{k-1}$ and $l = log_k\left(\frac{N_i(k-1)+1}{k}\right)$. If the tree is not full, we can consider $l = \left\lceil log_k\left(N_i\frac{k-1}{k}\right)\right\rceil$ (we also suppose that $N_i(k-1) \gg 1$). The total download time is then

$$T_{\text{PTree}}(b_i, C, k, N_i) = \frac{F}{b_i} + \frac{F}{b_i}\frac{k}{C}\left(\left\lceil log_k\left(N_i\frac{k-1}{k}\right)\right\rceil - 1\right) \tag{17}$$

where, we subtract 1 to the number of levels because the time to upload the first chunk to the leaves is included in the time to upload the whole file. If the class 2 is considered, the parameter k has to be substituted by the parameter s. Since there are no chunk exchanges between trees, the parameter f is zero. The difference with respect to (11) is only in the factor k of the first term.

Also for this case the optimization problem is solved independently for the two classes (see [4]), leading to optimal values $k = s = 3$.

Average download time. The average download time, with this architecture, is trivial: since all the peers belonging to the same class finish at the same time, the average for each class is equal to the total download time and the global average is $T_{\text{PTree}}^{\text{mean-indep}} = (N_1 \cdot T_{\text{PTree}}^{\text{Class1}} + N_2 \cdot T_{\text{PTree}}^{\text{Class2}})/(N_1 + N_2)$.

Amount of work. The number of copies distributed by the server is equal to the number of started roots for each class; since each roots download only a fraction of the file, the server uploads the whole file only twice. As regards fast and slow peers, each node uploads once.

2.3.2 Generous

Slow peers do not upload any chunk; they only download from fast peers; fast peers upload in parallel the chunk to k fast peers and to f slow peers. Here the bandwidth imposes more constraints: each fast peer, in fact, distributes a fraction k of the file F, while other k fast peers distribute the other fractions. The maximum bandwidth allowed toward a peer is then a fraction k of its capacity (since the peer receives up to k different chunks in parallel). Fast peers upload to f slow peers with bandwidth b_2/k for each slow peer. The remain bandwidth ($b_1^* = b_1 - f \cdot b_2/k$) is then used for fast peers.

Total download time. For the optimization problem, we have to find the best value for k and f (since there is no collaboration, $s = 0$). The optimum value k for the tree built by fast peers is independent from the fact that part of the bandwidth is dedicated to slow peers, so $k = 3$. The optimum value f must be chosen considering that we minimize the total download time if all the peers finish as much as possible together: to do so, all the N_1 generous fast peer must upload to all the N_2 slow peers, so $f = \left\lceil \frac{N_2}{N_1} \right\rceil$. Again, if f is sufficiently big, the bandwidth for class 1 (b_1^*) could be smaller than b_2 or, in the worst case, less than zero, and the total download time would be greater than in the Independent case (since class 2 is non-collaborative). We can conclude that $b_1 - f \cdot b_2/k$ should be greater than b_2, i.e. $f < \left(\frac{b_1}{b_2} - 1\right) \cdot k$; this means that, if $b_1 < (\frac{1}{k}\frac{N_2}{N_1} + 1)b_2$ the scheme is not applicable.

The total download time for class 1 is given by (17), where $b_i = b_1^*$. Looking at class 2, the download completes after F/b_2 rounds class 1 has reached the leaves of the fast tree, so

$$T_{\text{PTree}}^{\text{Class2}}(b_2, C, k, N_2) = \frac{F}{b_1^*}\frac{k}{C}\left\lceil log_k\left(N_1\frac{k-1}{k}\right)\right\rceil + \frac{F}{b_2}\ . \tag{18}$$

Average download time. The average download time depends on the total download time for each class and is equal to $(N_1 \cdot t_{\text{class 1}} + N_2 \cdot T_{\text{class 2}})/(N_1 + N_2)$.

Amount of work. The server uploads the whole file only once to class 1. As regards peers, each fast peer uploads $1 + \frac{1}{k}\frac{N_2}{N_1}$ times, whereas slow peers do not upload at all.

2.3.3 Generous, with Collaboration

In this case, slow peers are helped by fast peer, but here each served slow peer starts a new tree with degree s. We consider that this case, despite its complexity, does not introduce a great advantage with respect to the Generous case. In fact the impact in reducing f is on the smallest terms of (18) and the influence is not significant. We will verify via simulation this result.

In general, it is possible to say that any policy, with PTree architecture, does not greatly change the overall performance. In fact PTree obtains near-optimal results and any scheme can influence only the logarithmic term of (17) that is much smaller than the linear term.

2.3.4 Altruistic

Each fast peer, after uploading the content to k fast peers, stays on-line and starts to serve slow peers. In this case, we have to wait that all the fast peers complete the download; after that, $N_1\frac{b_1}{b_2}$ parallel copies are distributed to slow peers. We consider that this amount of copies is sufficient to complete the number of slow peers (so they upload only once).

Total download time. The time of slow peers to complete is

$$T_{\text{PTree}}^{\text{Class2}}(b_2, C, k, N_2) = \frac{F}{b_2} + T_{\text{PTree}}^{\text{Class1}}(b_1, C, k, N_1)\ . \tag{19}$$

In this case, before starting to serve slow peers, all the fast peers must finish: in fact, once the PTree structure is defined for the N_1 nodes, they complete to download all together. Even if a smaller number of fast peers is sufficient to serve slow peers (for instance, if $N_1 > N_2$), we have to wait for all the fast peers to complete.

Average download time. Again, the metric depends on the total download time for each class and is equal to $(N_1 \cdot T_{\text{PTree}}^{\text{Class1}} + N_2 \cdot T_{\text{PTree}}^{\text{Class2}})/(N_1 + N_2)$.

Amount of work. The server uploads the whole file only once to class 1. As regards peers, each fast peer uploads at most $1 + \frac{N_2}{N_1}$ times, whereas slow peers do not upload.

3 Analytic Results

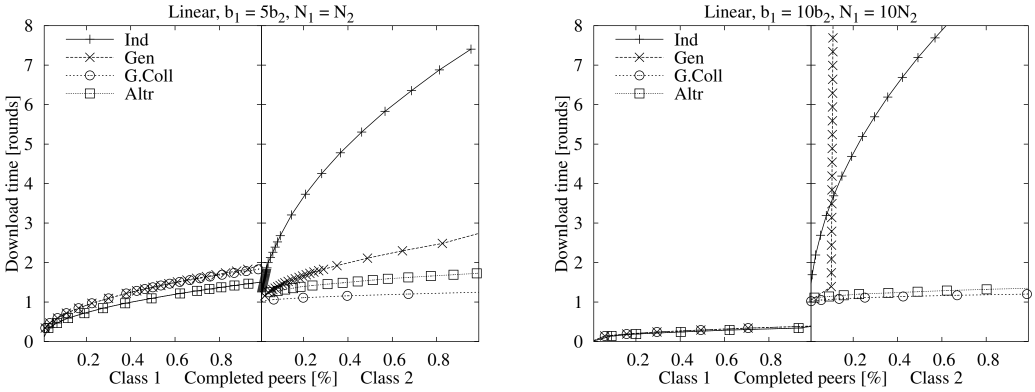
As numerical example we consider a set of bandwidth ratios, with two possible numbers of slow peers with respect to fast peers. In particular we focus on a total number of peers equal to 10^4: the analyzed cases correspond to (i) $N_2 = N_1 = 5000$ and (ii) $N_1 \simeq 900$ and $N_2 = 10 \cdot N_1 \simeq 9000$. We suppose a file size of 51.2 MB and a number of chunks equal to 200. We consider bandwidth b_2 such that $F/b_2 = 1$ round and bandwidth b_1 such that $F/b_1 = 0.5,\ 0.2,\ 0.1$ and 0.01. The last case corresponds to $b_1 = 100 \cdot b_2$ and can be considered as a limit to which the system evolves: with such a high bandwidth, in fact, fast peers complete the download very quickly and can help, with high capacity, low peers. The inferior limit that can be reached is a download time that tends to zero for fast peers and a download time that tends to 1 for slow peers (by construction $F/b_2 = 1$).

3.1 Total download time

Figure 10 shows how the system evolves in case of Linear architecture. Each plot shows the percentage of completed peers for each class with the four different schemes (Independent, Generous, ...). Left part shows the case $b_1 = 5b_2$ with $N_1 = N_2$, whereas right part shows the case of $b_1 = 10b_2$ with $N_2 = 10N_1$. It is possible to see that a helping policy greatly improves the class 2 performance, and the Generous with Collaboration reaches near optimal performance, regardless bandwidth ratios and number of peers ratio. In particular, in the left graph, class 2 finishes even before class 1. This is why a lot of slow chains are started almost contemporaneously (i.e., one slow chain every F/Cb_1, whereas only one fast chain every F/b_1 is created). Each of these slow chains, after the whole file is upload, produces a new slow peer every F/Cb_2, so in few slots of time (each slot is F/Cb_2) a lot of slow peers are reached.

In the case of $N_2 = 10N_1$, the download time with the Generous scheme grows rapidly since slow peers do not upload and, after fast peer have completed the download, only the server can upload to slow peers.

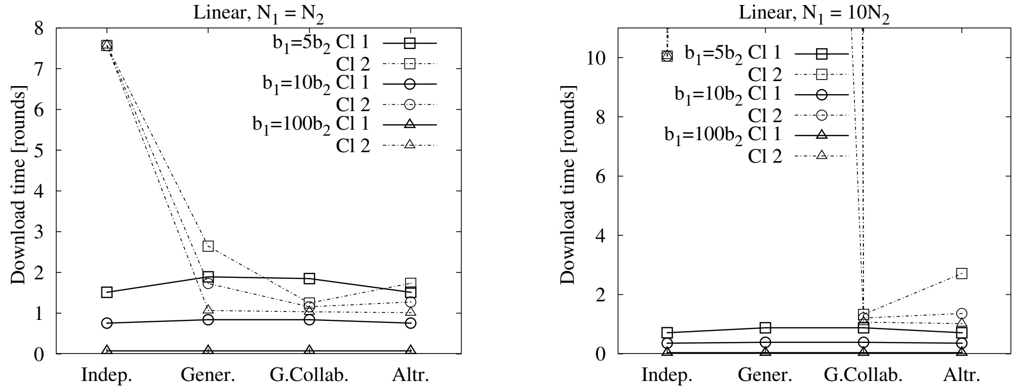
Figure 11 shows the complete set of results: for each scheme (reported on x-axis) it shows the total download time of each class with specific bandwidth ratios. Left side



(a) (b)

Figure 10: System evolution with Linear architecture: two examples

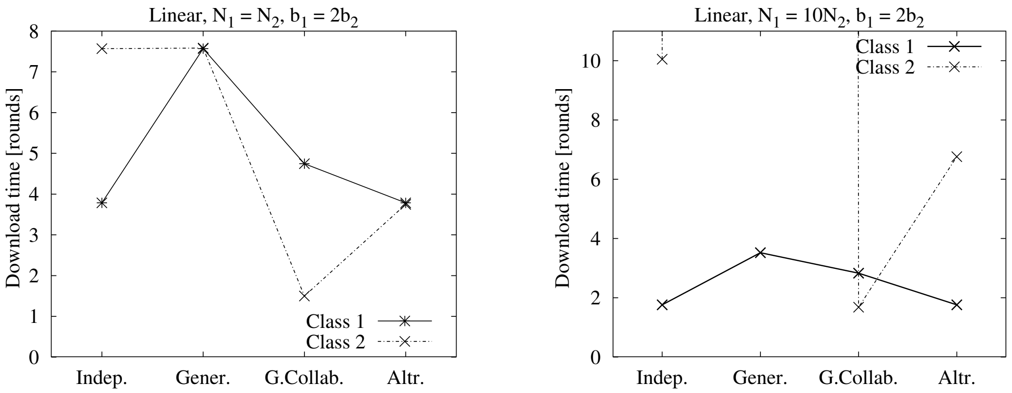
refers to $N_1 = N_2$ and right side to $N_2 = 10N_1$. The graphs show that a Generous with Collaboration scheme performs near optimally, with a little impact on class 1 performances. Points related to the Generous scheme, with $N_2 = 10N_1$, fall outside the plot area since the absolute value is very high due to the non collaborative behavior of slow peers and consequent work done only by the server. The Altruistic scheme performes worse than Generous with Collaboration since each fast peer starts to help slow peers after the download is complete, with a consequent delay.



(a) (b)

Figure 11: Total download time with Linear architecture achieved by each class with different schemes

The only exception to the results presented above is when $b_1 = 2b_2$ (Fig. 12). In fact, with a Generous scheme the bandwidth for class 1 becomes $b_1^* = b_1 - b_2 = b_2$. In case of collaboration, when class 2 has completed the download, class 1 can continue with full bandwidth, so its total download time decreases. Nevertheless, the Altruistic scheme, when $N_1 = N_2$, has better performance. The Generous with Collaboration scheme obtains near optimal performance only for class 2 or when the number of slow peers is much greater than the number of fast peers.



(a)

(b)

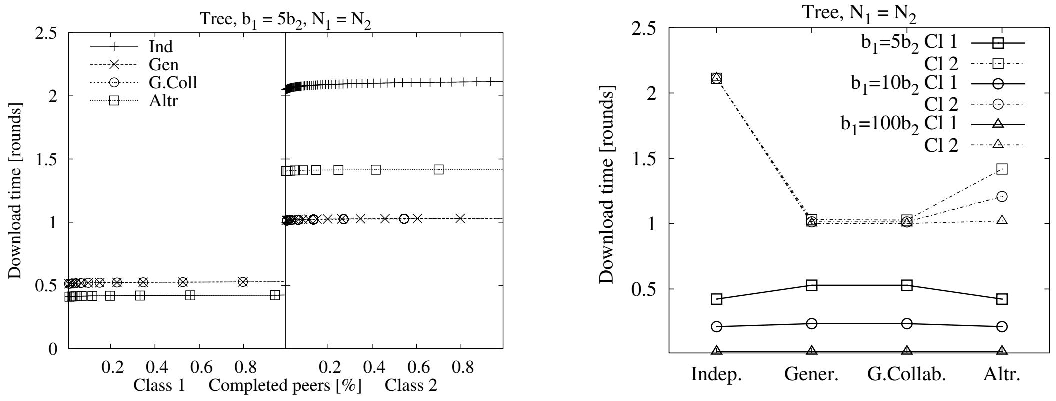
Figure 12: Total download time with Linear architecture in case of $b_1 = 2b_2$

Figure 13 shows the case with the Tree architecture. In the left graph the system evolution is plotted. The right graph reports the whole set of results. Here is shown only the case of $N_1 = N_2$, since higher number of peers ratios imposes higher bandwidth ratios, but the conclusions that can be drawn analyzing the graph remain the same. The Generous scheme performes optimally regardless bandwidth ratios. The collaboration, in this case, has no influence, since fast peers are able to help all the slow peers before the collaboration of slow peers can have effect.

Considering that the performance with the Generous scheme is near-optimal, the alternative Generous scheme we introduced in Sect. 2.2.2 (where only leaf nodes of the fast tree upload to slow peers) does not improve significantly the results. The only exception is when $b_1 = 2b_2$. The difference between the performance achievable by the two different Generous schemes is shown in Fig. 14. Class 1 performance improves, without affecting class 2.

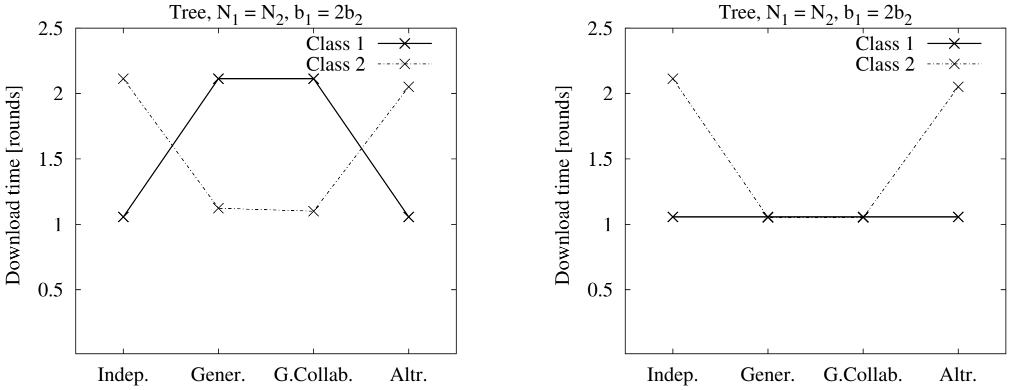
The Altruistic cases with Tree architecture (refer to the general case in Fig. 13) performs worse than Generous because each fast peer can start to help slow peers after it has completely download the content (at least kF/b_1), whereas in the Generous case it starts after it has download a single chunk (kF/Cb_1).

Figure 15 shows the results with PTree architecture. As we noted in the theoretical analysis, the different schemes do not influence the overall performance: in fact PTree obtain near-optimal performances for each class and a help from other class has little impact (only the logarithmic term is diminished, but this term is much smaller than the linear



(a) (b)

Figure 13: System evolution with Tree architecture (left) and Total Download Time for different cases (right)



(a) Generous scheme

(b) Alternative Generous scheme

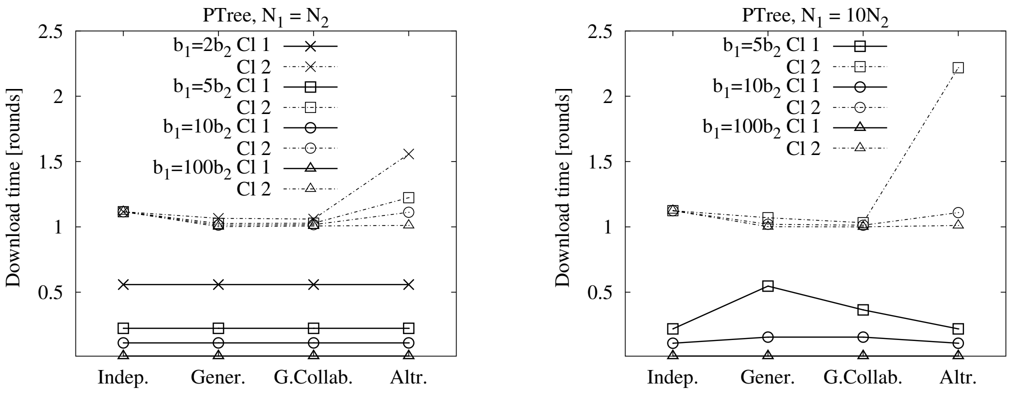
Figure 14: Total Download Time in case of Tree architecture, with $b_1 = 2b_2$

term). For instance, in case of $N_2 = 10N_1$, the Generous with Collaboration scheme improves the performance of class 1, but the impact on class 2 is not significant.

The Altruistic scheme give worse results than other scheme (even Independent) since slow peers have to wait the complete download of all fast peers[6].

After analyzing the single architectures, we can compare them finding some insights on the system. As regards Tree and PTree, thanks to the upload strategy adopted by the Tree architecture, the difference between the two architectures, when adopting a Gener-

[6] The Altruistic scheme imposes that a fast peer starts to help slow peers only after it has completed the download; with PTree all fast peers finish together.



(a) (b)

Figure 15: Total Download Time with PTree architecture using different schemes: $N_1 = N_2$ (left side) and $N_1 = 10N_2$ (right side)

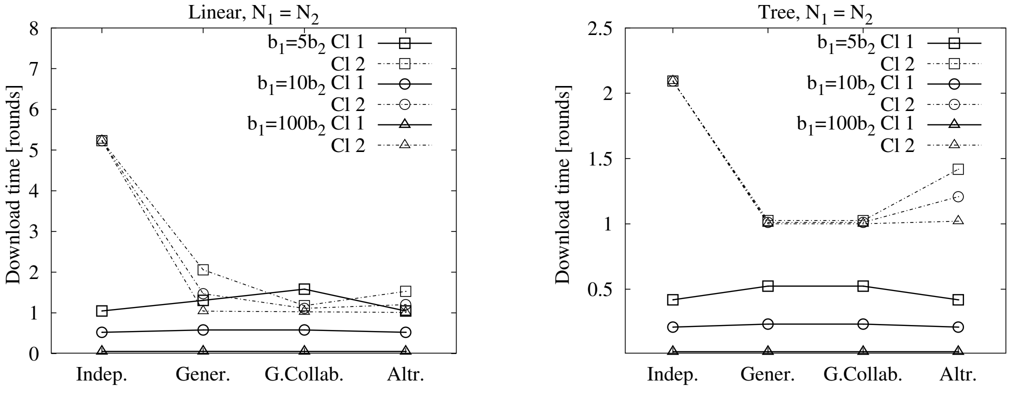
ous scheme, is very small, particularly when $b_1 > 2b_2$. The most interesting comparison is with the Linear architecture when the Generous with Collaboration scheme is adopted. When the number of slow peers is greater than the number of fast peers and the bandwidth of fast peers is sufficiently big, the differences between the performances of the two architectures are very small. This means that a very simple architecture with a very simple scheme (download from a fast or slow peer and upload to a slow peer) is as effective as a strongly structured architecture such as PTree. It is important to note that a Linear architecture with Generous with Collaboration scheme is equivalent to a mesh topology with indegree equal to one. The Tree architecture also can be built in a mesh topology by using simple rules.

3.2 Average download time

Figure 16 shows the average download time with Linear architecture, when $N_1 = N_2$. The behavior with the different schemes is similar to that we found with total download time. This result is true for all the other schemes, so the conclusions we have drawn before still hold.

3.3 Amount of work

Table 1 summarizes the amount of work for each architecture and scheme. Linear and PTree architectures maintain the amount of work of class 1 low. Tree architecture presents differences among peers of the same class: in fact interior nodes upload much more than leaf nodes. It is important to note that unfairness among peers does not mean that scheme is not applicable. In fact, there are situations where peers are not able to upload (e.g.,



(a)

(b)

Figure 16: Average Download Time with Linear (left) and Tree (right) architectures using different schemes ($N_1 = N_2$)

nodes behind a firewall or a NAT device). An unfair architecture or scheme can be used in such scenarios.

Table 1: Amount of work: summary table

Schemes	Server	Class 1	Class 2	Comments
Linear				
Independent	$\sum_i \frac{t_{\text{class i}}}{F/b_i}$	1	1	Last peers of each chain do not upload
Generous	$\frac{t_{\text{class 1}}}{F/b_1} + \max(0, N_1 + N_2)$	2	0	-
Gen. Collab.	$\frac{t_{\text{class 1}}}{F/b_1}$	≤ 2	≤ 1	-
Altruistic	$\frac{t_{\text{class 1}}}{F/b_1}$	$1 + N_2/N_1$	0	-
Tree				
Independent	$k + s$	k	s	Leaf nodes do not upload
Generous	k	$k + N_2/N_1$	0	-
Gen. Collab.	k	$\leq (k + N_2/N_1)$	1	-
Altruistic	k	$k + N_2/N_1$	0	-
PTree				
Independent	2	1	1	-
Generous	1	$1 + N_2/kN_1$	0	-
Gen. Collab.	1	$\leq (1 + N_2/kN_1)$	1	-
Altruistic	1	$1 + N_2/N_1$	0	-

4 Lesson Learned: Conclusions on the 2-Class Analysis

The heterogeneous case introduces a set of degrees of freedom in designing distribution policies; the interaction between the two classes can result in performance improvement even for simpler architectures and schemes. The different cases we analyzed show that, when we consider heterogeneity, slow peers may not negatively affect the system, provided that the difference in terms of capacities (and the associated constraints on number of peers) is sufficiently big (the ratio between fast peer capacity and slow peer one should be at least three-four). Collaborative distribution of a file can obtain as good performance as in the ideal distribution[7].

With Linear architecture, provided that the bandwidth of fast peer is sufficiently big, a Generous scheme with a collaborative behavior of slow peers is sufficient to obtain near-optimal performance.

With the Tree architecture, with the same hypothesis and scheme, we can also have near-optimal performance. The only constraint is that the capacity must be sufficiently big to serve all the N_2 slow peers.

PTree architecture is a scheme that reaches near optimal performances: this means that it is not possible to find a policy that significantly improves further the results. In general, we can conclude that the simpler the scheme is, the more improvement can be obtained with a collaborative distribution policy; the more complex the scheme is, the less the gain we have.

An important lesson learned is relative the Altruistic scheme: even if all the fast peers upload indefinitely to slow peers, the performance that can be obtain is worse than in a collaborative scheme. This means that collaboration of slow peers is necessary in order to achieve better results.

The simplicity of Linear and Tree architectures makes them optimal candidates for implementation in a mesh topology. For the Linear architecture is sufficient to impose that each fast peer must upload to a fast peer and to a slow peer and each slow peer must upload to another slow peer[8]. For the Tree architecture, is sufficient to impose that each fast peer has to upload to k fast peers in parallel and concurrently to f slow peers (f can be set by the user or estimated during the previous downloads); slow peers can or not upload to other slow peers. A problem that has to be solved is the global view of the network, i.e., peers have to upload choosing among their neighbors and not among all the peers in the network: the analysis of the performance with this limited view is left for future works.

[7] The ideal distribution is when there is a number of available copies in the network equal to the number of peers that wants to download the content, so that the time for downloading is F/b_i.

[8] The obvious constraints is that a peer can start to upload only to a peer that has not yet received any chunk.

5 Future Works

5.1 PTree for Groups

In this section we analyze a particular case where the altruistic behavior can have a great impact on performance. We remove the hypothesis that peers have a "global view" of all other peers: we suppose that peers are organized in clusters or groups and at least one peer of each group belongs to a "special group". For example, we can imagine a two-level hierarchy, where peers of the first level form a group (i.e., the special group) and each peer of the first level is the "head" of a group of peers of the second level. Consider that peers can have the same technological characteristics (e.g., the same bandwidth) and the division in levels is done only for organizing peers in a structure that does not require that peers know all the other peers: a peer that belongs to a group knows only the peers of its own group (except the peers that belongs to the special group, that knows more peers).

We want to show that with an altruistic behavior we can obtain good results even with this limited view.

We consider the homogeneous case, i.e. in the network there is only a single class of peers. At the end of the section we consider also the case of two classes. Suppose for simplicity that the tree is full and $F/b_1 = 1$. In 2.3 we found

$$T_{\text{PTree}}(C,k,N) = 1 + \frac{k}{C}\left(log_k\left(N_i\frac{k-1}{k}\right) - 1\right) . \tag{20}$$

If we start, instead of a single server, with S servers, then the number of peers can be divided in cluster of size N/S ad served independently. It is trivial to show that

$$T_{\text{PTree}}(C,k,N/S) = T_{\text{PTree}}(C,k,N) - \frac{k}{C}log_k S . \tag{21}$$

If we go back to the process of content distribution for the S server (the "special group" we have defined above) and we suppose that this distribution was performed with a PTree architecture, it is possible to find the time necessary to create such servers: it is simply $T_{\text{PTree}}(C,k,S)$. Then, the total time with this architecture is

$$\begin{aligned} T^{\#}_{\text{PTree}}(C,k,N) &= T_{\text{PTree}}(C,k,S) + T_{\text{PTree}}(C,k,N/S) = \\ &= 1 + \frac{k}{C}log_k\left(\frac{k-1}{k^2}\right) + T_{\text{PTree}}(C,k,N) = \\ &\simeq 1 + T_{\text{PTree}}(C,k,N) . \end{aligned} \tag{22}$$

It is trivial to show that in the general case this relation holds

$$T^{\#}_{\text{PTree}}(C,k,N) = F/b_i + T_{\text{PTree}}(C,k,N) . \tag{23}$$

This means that, if peers are organized in clusters such that at least one peer of each cluster belongs to the special group (formed by S peers that become servers), and if the peers belonging to the special group are altruistic, it is not necessary to have global view of the network; the content can be distributed as fast as PTree architecture, with a cost equal to a constant value that is the time for downloading the file F with a bandwidth b_i.

So only few altruistic peers can serve a lot of peers that have a limited view of the peers in the network. If we consider the case of two classes and suppose an altruistic behavior (so all the peers are altruistic, not only those belonging to the special group) the performance are dominated by fast peer and the results do not change.

5.2 Arbitrary Classes

The analysis made in Section 2 was an exhaustive study of the most interesting schemes that can be used with 2 classes. It is clear that such analysis can be indefinitely repeated considering more and more classes: nevertheless this implies only an application of the existing schemes, with solutions that become more and more cumbersome, and does not lead to a generalization of the problem.

As future works, it is possible to define a set of characteristics of the system and to try to solve the problem of heterogeneous sources statistically, instead of deterministically.

We assume the presence of an arbitrary number of classes: each class have bandwidth b_i with a probability p_i. We assume that the distribution of the probability is known. We assume also that all the peers know all the other peers and their bandwidth: these implies that peer belonging to a specific class can be organized in one of the three possible architecture: Linear, Tree or PTree. Each peer can decide to use a fraction of its capacity to help other classes. The scope of this analysis is to find general guidelines for deciding the best way to help other classes.

5.3 Unstructured Peers

In this section we collect some preliminary observations relative to the case when the hypothesis that peers bandwidths are known is removed. This situation happens when peers can be involved in more than single file exchange, so only a fraction of the bandwidth is dedicated to the file exchange we are interested in.

We assign a probability p_i that a peer has bandwidth b_i; we assume that the distribution of b_i is equal for all the peers and the b_i are statistically independent variables. The distribution of b_i summarizes the fact that in the network peers dedicate part of the bandwidth for file distribution and also the fact that in the network there could be peers belonging to different classes. Table 2 shows the distribution of connection bandwidths that was measured in [reference].

As a first analysis we suppose that if a peer starts to upload to a peer with less bandwidth (so there is a fraction of the upload bandwidth available), the number of parallel uploads can not change and remains constant, with a consequent waste of upload capacity: for instance, in the Linear case, if a peer starts to upload to a slower peer, it can not use the left capacity to start a parallel upload.

Table 2: Bandwidth distribution

Bandwidths	%
56 Kbps	6%
DSL	23%
Cable	64%
T1	5%
T3	2%

5.3.1 Linear Distribution

Peers are organized in Linear chains, i.e. a peer can download from exactly one peer and it can upload to exactly one peer, provided that, when the upload starts, the peer has not yet received any chunk. The server starts to distribute the file at maximum rate: we suppose that all the heads of the chains are fast peer with full bandwidth available; this means that a new chain is started every F/b_{high} seconds. We observe a single chain and try to find the time necessary to reach N_α peers: it is clear that as soon as there is a peer with low bandwidth all the following peers in the chain will distribute the content with the rate imposed by the slow peer. The more the chain gets long, the more the probability of finding a peer with low bandwidth increases.

Consider a node i in the chain: the bandwidth b_i^t used to upload the content is the minimum between the bandwidth of the node from which the peer is downloading and the capacity of the peer itself (see Fig. 24); in formulas $b_i^t = \min(b_{i-1}^t, b_i)$. The bandwidth of the node $i+1$ to which the peer uploads is not taken into account since it is considered in the analysis of the node $i+1$. The cumulative distribution of the variable b_i^t is then a combination of the two variables b_{i-1}^t and b_i and it is equal to

$$F_{b_i^t}(x) = F_{b_{i-1}^t}(x) + F_{b_i}(x) - F_{b_{i-1}^t}(x)F_{b_i}(x)\ . \tag{24}$$

Calculating iteratively the cumulative distribution with (24) it is possible to find the final distribution of b_{final} of the last node and then the total time necessary to download the content. The only quantity that has to be find is the time necessary to reach the last peer: this is equal to the sum of the independent times[9] necessary to reach each peer of the chain and the probability density function is the convolution of the probability density function of each b_i^t.

The calculation of the distribution of a chain can be repeated for all the chains started by the server and then it is possible to find the mean time necessary to complete the download of N peers.

[9] We suppose that the variables b_i are independent and consequently t_i.

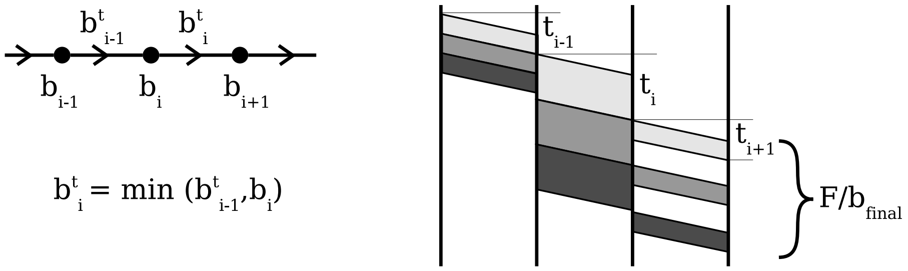


Figure 17: System model

5.3.2 Tree Distribution

In this case peers are organized in trees, i.e. a peer can download from exactly one peer and can upload to k peers, provided that, when the upload starts, the peers has not yet received any chunk. The server initially distribute the file to k peers that we consider fast peers. In this case as soon as a peer with low bandwidth is reached, the entire tree generated by the peer will evolve with a low bandwidth. The impact on the file distribution is then significant. The analysis of this case is left as future works.

References

[1] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, "Wide-area cooperative storage with CFS," in Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 01), Chateau Lake Lou, Banff, Canada, Oct. 2001

[2] Grid Computing, http://www.gridcomputing.com/

[3] B. Cohen, "Incentives build robustness in BitTorrent," May 2003

[4] E.W. Biersack, P. Rodriguez, P. Felber, "Performance Analysis of Peer-to-Peer Networks for File Distribution." Proceedings of the 5th International Workshop on Quality of future Internet Services (QofIS'04), Barcelona, Spain, Sept. 2004.