Peer-to-peer Distribution Architectures providing Uniform Download Rates

Marc Schiely, Pascal Felber

Computer Science Department
University of Neuchâtel
CH-2007, Neuchâtel, Switzerland
marc.schiely@unine.ch, pascal.felber@unine.ch

Abstract. Peer-to-peer (P2P) networks have proved to be a powerful and highly scalable alternative to traditional client-server architectures for content distribution. They offer the technical means to efficiently distribute data to millions of clients simultaneously with very low infrastructural cost. Previous studies of content distribution architectures have primarily focused on homogeneous systems where the bandwidth capacities of all peers are similar, or simple heterogeneous scenarios where different classes of peers with symmetric bandwidth try to minimize the average download duration. In this paper, we study the problem of content distribution under the assumption that peers have heterogeneous and asymmetric bandwidth (typical for ADSL connections), with the objective to provide uniform download rates to all peers—a desirable property for distributing streaming content. We discuss architectures that fulfill this goal and achieve optimal utilization of the aggregate uplink capacity of the peers. We develop analytical models that provide insight on their performance in various configurations, and we compare them to architectures with non-uniform rates. Our results indicate that heterogeneous and asymmetric peers can achieve uniform download rates with little additional complexity and no performance penalty.

1 Introduction

The distribution of large or streaming content remains a challenging problem in today's Internet. A single source quickly becomes saturated when the number of clients requesting the content grows, which leads to degradation or loss of service. Solutions based on content delivery networks (CDNs) are prohibitively expensive and rather static in nature, while protocols like IP multicast suffer from several flaws and are not widely deployed.

P2P systems, in which peer computers form a cooperative network and share their resources, offer a promising alternative for content distribution. They reduce the load of the primary servers by leveraging the bandwidth of the peers, those receiving part of the content providing it to others. Their low cost, inherent scalability, and resiliency to "flash crowds" (a huge and sudden surge of request traffic that usually leads to the collapse of the affected server) make such systems very attractive for large scale deployments.

This work focuses on P2P architectures designed for distributing content among large populations of clients. Unlike previous studies, we assume that the peers have heterogeneous and asymmetric bandwidth (typical for ADSL connections) and we aim at providing a uniform download rate to each of them. This property is crucial for applications like media streaming, for which users expect an uninterrupted stream of data at a constant rate.

We consider simple models with two classes of peers that differ in their uplink capacities. We study several architectures that achieve optimal utilization of the aggregate uplink capacity of the system and share it equally between all the peers. It obviously follows that fast peers must share more bandwidth than they receive, but we can balance this unfairness by placing them nearer to the source for increased reliability and shorter latency.

We develop analytical models that provide interesting insight on the performance of content distribution architectures with uniform download rates in various configurations. We compare them with other architectures providing non-uniform rates and we conclude that uniformity can be achieved with little additional complexity and no performance penalty.

The rest of the paper is organized as follows: We first discuss related work in Section 2 and we present the system model in Section 3. Then, we analyze three different architectures providing uniform download rates in Section 4 and compare them in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

There exist two main approaches for dealing with differences in uplink bandwidth in overlay multicast systems. Narada [1], CollectCast [2] and GnuStream [3] use bandwidth measurements to improve the overlay structure by dynamically replacing links. In contrast Scattercast [4], SplitStream [5], Overcast [6] and ALMI [7] use degree-constraint structures to deal with heterogeneity. If a peer's degree is saturated when a new peer wants to connect, then some reorganization needs to take place. CoopNet [8] uses both of these techniques. It deploys multiple parallel trees and reorganizes them based on performance feedbacks.

All of these systems do not try to uniformly distribute the download rate to all peers. Instead, they send distinct streams at different rates, or they consider bounded streams and use buffers to deal with timing problems. Our goal is to minimize these buffer requirements by evening out the download rate at all peers.

In [9], the authors investigate the impact of heterogeneous uplink bandwidth capacities on Scribe [10]. Their experiments show that heterogeneity may create distribution trees with high depths, which is not desirable. After proposing several ways to address the problem they conclude that heterogeneity in DHT-based multicast protocols remains a challenging open problem.

Analytical models have been proposed for peers with homogeneous bandwidth capacities [11, 12], as well as and for heterogeneous peers but for non-uniform download rates [13]. Different architectures for homogeneous and heterogeneous bandwidth constraints are analyzed. In contrast to this work, the

authors make the assumption that the downlink and uplink capacities are symmetric and do not consider uniform download rates.

To the best of our knowledge, no analytical models have been proposed to study P2P content distribution architectures providing uniform download rates to heterogeneous peers with asymmetric bandwidths.

3 System Model and Definitions

For the rest of this paper we use the following model. We assume that nodes in the network have different upload capacities. We analyze content distribution architectures with two classes of nodes, referred to as *fast* and *slow* peers according to their upload bandwidth. All nodes in a class have the same bandwidth. The data stream is sent by a single source which has the same bandwidth as fast nodes. To simplify the analysis, we assume that the source receives the data at the same uniform rate as the other peers before distributing it within the content distribution network. We shall ignore latency in our model.

As is the case for typical ADSL connection, we assume that the slow peers are essentially limited by their uplink capacity and have sufficient download bandwidth to receive the data at the same uniform rate as the other peers.[1] We consider N_f fast peers in class F with upload bandwidth B_f and N_s slow peers in class S with upload bandwidth B_s ($B_f > B_s$). For the sake of simplicity, we assume in our analysis that $B_s = \frac{B_f}{k}$ with k being an integer value. The total number of peers is $N = N_f + N_s$.

We analyze the behavior of different architectures when transmitting a large content. We assume that the file being transmitted is split into C chunks that can be sent independently: as soon as a peer has received a chunk, it can start sending it to another peer. We consider one unit of time to be the time necessary for transmitting the whole content at the uniform rate r that is provided to all peers. Each chunk is thus received in $\frac{1}{C}$ unit of time. For clarity, we shall describe the different architectures with the assumption that we transmit the whole file at once and we shall introduce chunks later in the analysis. As total download time is a function of the number of chunks, our main objective of supporting streaming data corresponds to situations where $C \to \infty$.

A peer may receive chunks from the source via different paths. For instance, in the case of SplitStream [5], the source splits the content into several layers and sends each of them along distinct trees spanning all the nodes. Two chunks sent *at the same time* by the source may thus traverse a different number of peers and be received at different times. This implies that each peer may have to buffer some chunks until all of those sent at the same time have been received. We compute δ_T as the maximal difference in distance between a peer and the closest common node along the paths to the source via distinct incoming links. This value indicates the buffer space needed at the peer. For instance, in Figure 1, the first node of the right chain receives chunks from the source in 1 (directly),

[1] As we shall see, this rate is no higher than the uplink capacity of the fast peers.

2 (via one peer), and 3 (via two peers) units of time and we have $\delta_T = 3 - 1 = 2$. Clearly, small values of δ_T are desirable and we shall also compare the different architectures with respect to this property.

Uniform Rate. As previously mentioned, our goal is to provide the same download rate to all peers in the network. Obviously, the maximal rate r that can be achieved corresponds to the aggregate upload bandwidth of all nodes divided by the number of peers ($B_s < r < B_f$). It is easy to see that a tree cannot be used to fulfill this goal because a slow node does not have enough upload bandwidth to serve even a single other peer at rate $r > B_s$.

A trivial approach is to form chains of peers, in which a combination of slow and fast peers team up and share their bandwidths at each level of the chain. Figure 1 shows such an architecture with 50% fast nodes and 50% slow nodes ($N_f = N_s = \frac{N}{2}$), and slow nodes having half of the upload bandwidth of fast nodes ($B_s = \frac{B_f}{2}$). The source is the topmost node and the numbers show the transmission rate on the corresponding link, as a fraction of B_f. Fast nodes are displayed in gray. The time units indicated in the figure do not explicitly take chunks into account: at $t = 1$, the second peer in the left chain has received the content at rate $\frac{3}{4}$; at $t = 3$, the first peer in the right chain has received the content via three links, each at rate $\frac{1}{4}$; etc. All time units should be divided by C when considering chunks. The download rate r is calculated as:

$$r = \frac{1}{N}(\frac{N}{2}B_f + \frac{N}{2}\frac{B_f}{2}) = \frac{3}{4}B_f$$

We can observe that an unused upload bandwidth of $\frac{3}{4}B_f$ remains because the source does not download any content. We shall ignore it in the rest of the paper.

If we generalize the upload bandwidth of the slow peers to a fraction of the upload bandwidth of the fast peers $B_s = \frac{B_f}{k}$ and compute the download rate r for a scenario where $N_f = N_s = \frac{N}{2}$, we obtain:

$$r = \frac{1}{N}(\frac{N}{2}B_f + \frac{N}{2}\frac{B_f}{k}) = \frac{k+1}{2k}B_f$$

We now relax the assumptions on the distribution of fast and slow nodes. If the number of fast peers is N_f, then the number of slow peers is $N_s = N - N_f$. Again the upload bandwidth of the slow peers is a fraction k of the upload bandwidth of the fast peers. The optimal download rate is then:

$$\begin{aligned} r &= \frac{1}{N}(N_f B_f + (N - N_f)\frac{B_f}{k}) \\ &= (\frac{N_f}{N} + \frac{1}{k}(1 - \frac{N_f}{N}))B_f \end{aligned} \quad (1)$$

If slow peers do not serve any content, i.e., $k \to \infty$, then Equation (1) becomes:

$$\lim_{k\to\infty}(\frac{N_f}{N}+\frac{1}{k}(1-\frac{N_f}{N}))B_f = B_f\frac{N_f}{N}$$

In a scenario where $N_f = N_s$ this leads to a binary tree where the slow nodes are the leaves and the fast nodes are the inner nodes, each serving two other nodes at rate $r = \frac{B_f}{2}$ (as studied in [11]).

In the rest of the paper, unless explicitly mentioned, we consider equal populations of slow and fast peers ($N_s = N_f$).

4 A Study of Three Architectures

We now study and compare three different architectures that provide a uniform download rate to all peers.

4.1 Linear Chain Architecture

The first architecture considered in this paper forks several independent chains of peers that distribute content in parallel. The chains are constructed in three phases.

Phase 1 - Growing phase. The objective of the growing phase is to serve several peers (say m) in parallel starting from a single source. Obviously, such an expansion can only be achieved by fast peers, as they have more upload capacity than the target download rate r. Using this free capacity allows us to build the service capacity mr necessary to serve m peers in parallel.

Informally, the growing phase proceeds as follows. The first fast node (the source) starts a chain by serving one other fast peer with rate r. The remaining bandwidth $B_f - r$ will be used further down the chain. The second fast peer again serves another fast peer with rate r, which also leaves it with $B_f - r$ remaining bandwidth. This process continues until the sum of the remaining bandwidths of the first p fast nodes is sufficient to serve another peer, i.e., $p(B_f - r) \geq r$. Given that $B_s = \frac{1}{k}B_f$, p can be computed as:

$$p = \left\lceil \frac{k+1}{k-1} \right\rceil$$

In the formula above, depending on the value of k, some bandwidth may be lost in the integer conversion. This can be avoided by expanding to k nodes at once. The number of peers p_k necessary for this expansion can be computed by solving $p_k(B_f - r) = r(k-1)$, which gives:

$$p_k = k + 1$$

In the rest of the paper, we shall assume expansions to k chains using p_k peers (instead of 2 chains using p peers). Each fast peer can in turn fork another

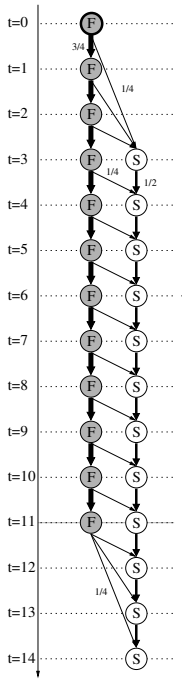

Fig. 1. Linear chains with one expansion step and $k = 2$ ($B_s = \frac{B_f}{2}$).

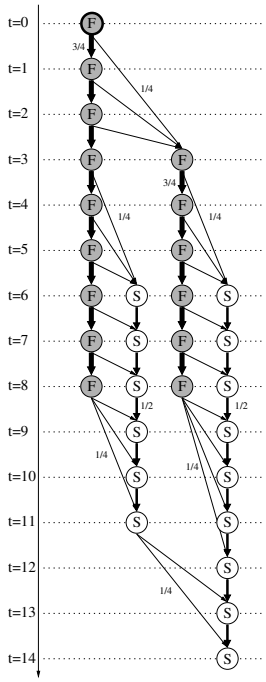

Fig. 2. Linear chains with two expansion steps and $k = 2$ ($B_s = \frac{B_f}{2}$).

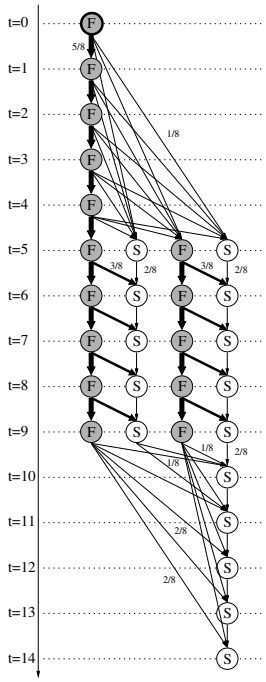

Fig. 3. Linear chains with one expansion step and $k = 4$ ($B_s = \frac{B_f}{4}$).

k chains with the help of $p_k - 1$ other fast peers. By repeating this process, the number of chains can be multiplied by k every iteration. Each expansion obviously requires p_k units of time. Examples with $k = 2$ ($r = \frac{3}{4}B_f$) and $k = 4$ ($r = \frac{5}{8}B_f$) are shown in Figures 1, 2, and 3. It is important to note that the peers are organized as a directed acyclic graph (DAG).

Phase 2 - Parallel phase. The parallel phase starts when the growing phase has finished its expansion to m peers. It constructs two sets of $\frac{m}{2}$ linear chains, composed respectively of fast and slow peers. Each chain of slow peers is combined with a chain of fast peers. A slow peer serves its successor at rate B_f/k. A fast peer serves its successor at rate r and the next slow peer in the companion chain at rate $B_f - r$. Thus, each peer is served at rate r. Phase 2 proceeds until all fast peers are being served (see Figures 1, 2, and 3).

Phase 3 - Shrinking phase. In the last phase, we are left with a set of slow peers to serve at rate r. As a slow peer cannot serve another peer by itself, the bandwidth of several peers must be combined, which leads to shrinking down the number of parallel chains. This phase is almost symmetrical to the growing

phase, in that we can serve p_k slow peers from each set of k chains. We repeat this process until all slow peers have been served (see Figures 1, 2, and 3).

Analysis. We can easily notice that delays of $\delta_T = k$ are encountered during the growing phase. The case of the shrinking phase is more subtle, as δ_T grows larger if we keep it perfectly symmetric to the growing phase. By allowing some asymmetry, we can both bound the delays by the same value $\delta_T = k$ and reduce the total length of the shrinking phase.

We now compute the number of peers that can be served within a given time interval. After p_k steps, k peers can start again another chain. If we define s as the number of expansion steps, we can calculate the number of peers in the first phase N_1 to be:

$$N_1 = \sum_{i=0}^{s-1} k^i p_k = p_k \frac{k^s - 1}{k - 1}$$

The shrinking phase is built in a symmetric manner. Therefore the number of nodes N_3 in the third phase is the same as in the growing phase: $N_3 = N_1$. Given the constraint that $N_1 + N_3 \leq N$, the maximal value of s is:

$$s_{max} = log_k \left(N \frac{k-1}{2p_k} + 1 \right)$$

The number of nodes N_2 that can be served in phase 2 in a given time interval T is:

$$N_2 = k^s (T - 2sp_k + 1)$$

Indeed, there are k^s parallel nodes and phase 2 lasts for the given time interval minus the duration of the growing and shrinking phases. The number of peers served in a time interval T with s growing steps ($1 \leq s \leq \lfloor s_{max} \rfloor$) is then:

$$N(T, s, k) = 2p_k \frac{k^s - 1}{k - 1} + k^s (T - 2sp_k + 1)$$

We observe that the number of peers served in a given time interval grows with s, thus producing more efficient content distribution architectures (compare $N(14, 1, 2) = 24$ in Figure 1 and $N(14, 2, 2) = 30$ in Figure 3).

Solving the equation for T gives the number of units of time necessary to serve N peers:

$$T(N, s, k) = \frac{N(k-1) - 2p_k(k^s - 1)}{k^s (k - 1)} + 2sp_k - 1 \tag{2}$$

Assuming that the content is split into chunks, the total download time for the complete file is then $1 + \frac{1}{C} T(N, s, k)$, i.e., the time necessary to transmit the whole file at rate r plus the propagation time of the chunks through the content distribution network. Using Equation (2) leads to:

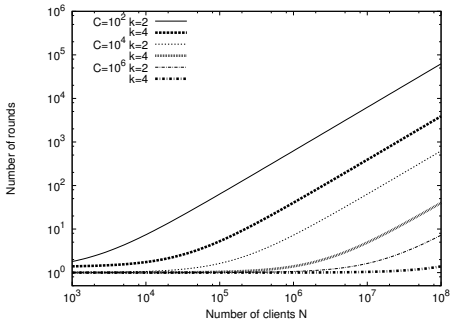

Fig. 4. Download time of the linear chain architecture for different values of k and C ($s = 4$).

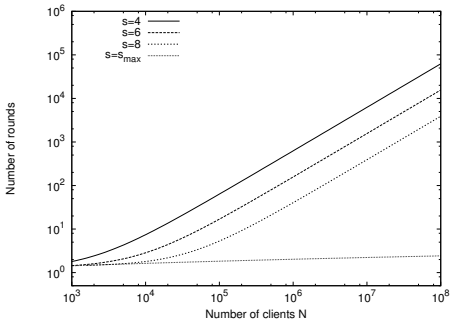

Fig. 5. Download time of the linear chain architecture for different values of s ($k = 2$, $C = 10^2$).

$$T(N, s, k, C) = 1 + \frac{1}{C}\left(\frac{N(k-1) - 2p_k(k^s - 1)}{k^s(k-1)} + 2sp_k - 1\right)$$

Figure 4 shows the time necessary to complete the download with the linear chain architecture for different values of k and C. We observe that performance improves with larger numbers of chunks, because all peers can be active most of the time. In contrast, with few chunks only a fraction of the peers will be uploading at any point in time, while the others have either already forwarded the entire file or not yet received a single chunk. Therefore, the value of k, which influences the depth of the content distribution architecture, has more impact on performance when the number of chunks is small. We notice indeed that the download times start degrading earlier with small values of k because they yield deeper architectures.

Figure 5 compares the download times for different values of s (the value s_{max} corresponds to the maximal number of expansion possible with the given peer population). As expected, performance improves with higher values of s because they produce flatter architectures. The optimal value s_{max} exhibits extremely good scalability.

4.2 Mesh Architecture

The linear chains architecture can be improved in several ways if we allow peers to be organized as a directed graph with cycles. We can reduce the duration of the growing phase and thus the length of the paths (and consequently the latency); we can simplify network management by only using connections with identical bandwidth capacities; and we can limit the size of buffers at each peer to a constant value.

The resulting mesh architecture is shown in Figure 6 (for $k = 2$ and one expansion step) and Figure 7 (for a general value of k and two expansion steps). A node does not only receive data from its parent, but also from its siblings. The source has $2k$ fast peers as children and sends data at rate $\frac{B_f}{2k}$ to each of them; the remaining bandwidth $\frac{B_f}{2}$ is provided by their siblings. The first-level fast peers together serve k^2 children with their remaining bandwidth of $\frac{B_f}{2}$; again, the remaining bandwidth $\frac{k-1}{2k}B_f$ is provided by the siblings. Second-level peers have enough bandwidth to completely serve k^2 children. Each third-level child can in turn expand to k^2 peers in three steps.

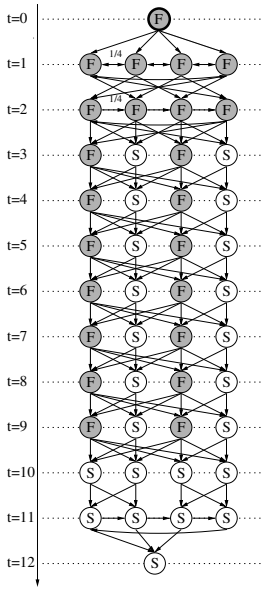

Fig. 6. Mesh with one expansion step and $k = 2$ ($B_s = \frac{B_f}{2}$).

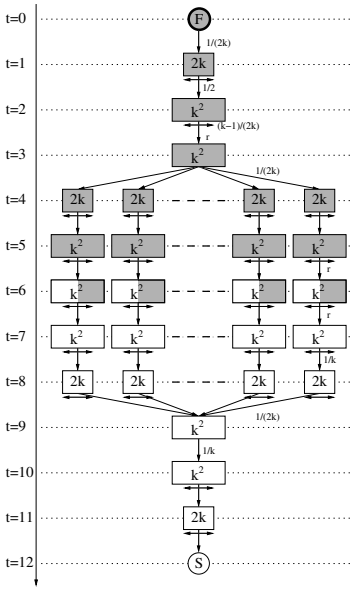

Fig. 7. Mesh with two expansion steps and any k ($B_s = \frac{B_f}{k}$).

As in the previous architecture, one can build linear chains after the expansion phase before reducing the architecture to one peer. The shrinking phase is symmetric to the growing phase, as shown in Figure 6.

Using only connections with identical rate $\frac{B_f}{2k}$ simplifies significantly the management of the architecture. The throughput is controlled by the source and peers only differ in their number of outgoing connections: the outdegree is always $2k$ for fast nodes and 2 for slow nodes. All peers have an indegree of $k + 1$.

Analysis. One can note in Figure 6 that the first level fast peers receive chunks from the source at $t = 1$ and from their sibling at $t = 2$; similarly, second level

peers receive chunks at $t = 2$ and $t = 3$; on the third level, all chunks are received simultaneously at $t = 3$. A similar observation can be made with the shrinking phase and it follows that constant delays of $\delta_T = 1$ are encountered in this content distribution architecture.

For computing the number of nodes which can be served in time T we again analyze the three phases. As we have seen, a fast peer can expand to k^2 peers in three units of time with the help of $2k + k^2$ other fast peers. If we define s to be the number of expansion steps, then the number of peers served in the first phase is:

$$N_1 = 1 + (2k + 2k^2) \sum_{i=0}^{s-1} k^{2i} = 1 + 2k \frac{k^{2s} - 1}{k - 1}$$

The shrinking phase again is symmetric in the number of nodes so the number of nodes in the third phase N_3 is equal to N_1, thus $N_3 = N_1$. Given the constraint that $N_1 + N_3 \leq N$ we can compute the maximal value of s:

$$s_{max} = \frac{1}{2} log_k \left(\frac{(N-2)(k-1)}{4k} + 1 \right)$$

In phase 2, $k^2 k^{2(s-1)}$ parallel nodes can be served in the remaining time $T - 6s - 1$. In total the number of peers served within T units of time for a given number of s expansion steps $1 \leq s \leq \lfloor s_{max} \rfloor$ is then:

$$N(T, s, k) = 2 + 4k \frac{k^{2s} - 1}{k - 1} + k^{2s}(T - 6s - 1)$$

Solving the equation for T and introducing the number of chunks C gives:

$$T(N, s, k, C) = 1 + \frac{1}{C} \left(\frac{1}{k^{2s}} \left(N - 2 - 4k \frac{k^{2s} - 1}{k - 1} \right) + 6s + 1 \right)$$

Figure 8 shows the time necessary to complete the download with the use of the mesh architecture for different values of C and k. As expected, the download times follow the same general shape as for the linear chains architecture in Figure 4 but performance is significantly improved due to the faster expansion of the mesh architecture. We can observe in Figure 9 that a higher number of expansion steps s also produces flatter architectures and therefore reduces the download time. The maximal expansion for a given peer population s_{max} yields the best download times, which is almost constant, independent of the population size.

4.3 Parallel Trees

The third architecture studied in this paper consists in constructing multiple trees spanning all the nodes and sending a separate part of the content in parallel to each tree similarly to SplitStream [5] and PTree^k [11] (as $N_f = N_s$, we shall

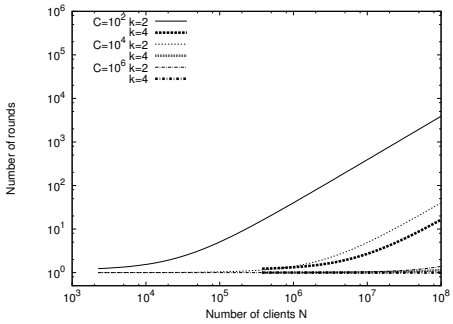

Fig. 8. Download time of the mesh architecture for different values of C ($k = 2$, $s = 4$).

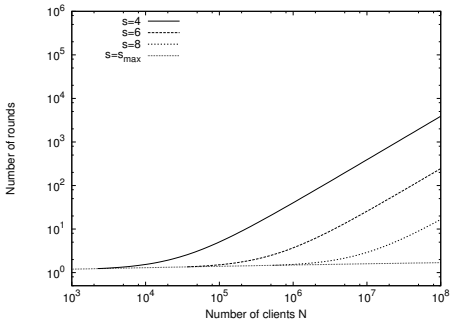

Fig. 9. Download time of the mesh architecture for different values of s ($k = 2$, $C = 10^2$).

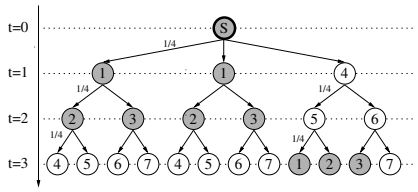

Fig. 10. Parallel trees with $N = 8$ and $k = 2$ ($B_s = \frac{B_f}{2}$).

use binary trees). If we construct $k+1$ trees that distribute content at rate $\frac{B_f}{2k}$, then every peer will receive data at the same uniform rate r.

We construct parallel trees by placing each fast peer (except the source) as interior node in k trees. Fast nodes will thus serve $2k$ other peers at rate $\frac{B_f}{2k}k$ (i.e., at aggregate rate B_f). The slow nodes are placed as interior nodes in a single tree and must thus serve two other nodes at rate $\frac{B_f}{2k}$ (i.e., at aggregate rate $\frac{B_f}{k}$). As the number of leaves in a complete binary tree is equal to the number of interior nodes plus one and the source is a fast node, the constraint $N_f = N_s$ is met. Figure 10 illustrates the parallel tree architecture (peers are numbered for clarity). Note that every peer except the source appears in all trees.

Analysis. We first need to determine the depth d of the trees. At each level i in the tree, we have 2^i nodes (the root is at level 0). Thus, the number of nodes in a binary tree of depth d is $\sum_{i=0}^{d} 2^i = 2^{d+1} - 1$. Considering the special role of the source, the $N - 1$ remaining nodes can be placed in parallel trees of depth $d = \lfloor log_2(N-1) \rfloor$.

It follows from the construction of the trees that delays of $\delta_T = \lfloor log_2(N-1) \rfloor$ are encountered in this content distribution architecture. Delays grow with the number of peers, in contrast to the other architectures studied in this paper.

The number of nodes that can be served by the parallel tree architecture in a given time interval T can be computed as follows (the first term represents the source):

$$N(T) = 1 + \sum_{i=0}^{T-1} 2^i = 2^T$$

Solving this equation to T and introducing the number of chunks C leads to the time used to distribute a file to all nodes:

$$T(C, N) = 1 + \frac{1}{C} \lceil log_2 N \rceil$$

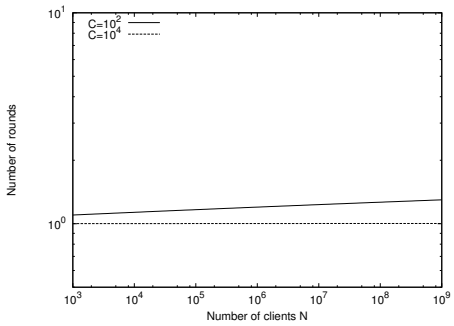

Fig. 11. Download time for the parallel trees architecture for different values of C.

Figure 11 shows the time necessary to complete the download with the parallel tree architecture for two values of C (improvements become unnoticeable when C grows larger). As the download time is a function of the depth of the trees, which increases logarithmically with the number of peers, performance degrades only slowly with the population size.

5 Comparative Analysis

In this section we compare the three architectures presented in this paper with the linear chain architecture analyzed in [13] (referred to as *Linear*). In contrast to our architectures, in *Linear* the peers have symmetric bandwidth capacities. The peers are organized in separate chains according to their bandwidth capacity

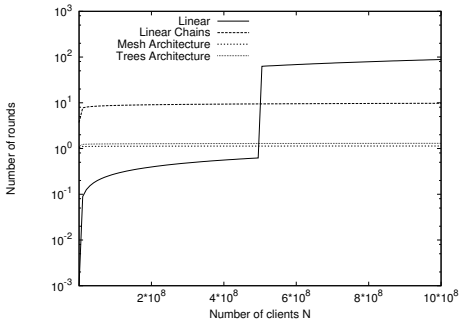

Fig. 12. Download time for different architectures with $k = 100$, $C = 100$ and $s = s_{max}$. *Linear* shows the completion times for a population of 10^9 peers with symmetric bandwidth.

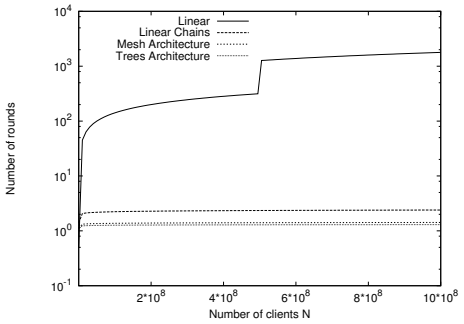

Fig. 13. Download time for different architectures with $k = 4$, $C = 100$ and $s = s_{max}$. *Linear* shows the completion times for a population of 10^9 peers with symmetric bandwidth.

and there is no cooperation between fast and slow nodes. Fast peers can therefore finish the download faster.

As we can see in Figure 12, this difference leads to a stepwise function with the fast nodes completing their download faster than the slow nodes ($N_f = N_s$). In contrast, the uniform architectures all scale well and yield an almost constant download rate independent of the population size. As expected, uniform linear chains are less efficient than the mesh and parallel tree architectures due to the longer paths.

In Figure 13 we can observe that with a smaller difference between fast and slow peers (lower value of k) the download time of *Linear* grows, whereas it decreases for the linear chains and the mesh architecture (remember that a unit of time is defined as a function of the uniform rate r). We can further see that the mesh architecture performs slightly better than parallel trees in Figure 12, unlike in Figure 13. This is due to the fact that the mesh architecture expands as a function of k^{2s} whereas the expansion of parallel trees does not depend on k. Thus the mesh will grow faster when k is large. Higher values of C do not produce interesting results as the difference between the various architectures quickly becomes unnoticeable.

6 Conclusion

Content distribution is an important problem for many distributed applications deployed in the Internet. Cooperative techniques based on peer-to-peer networks offer the technical capabilities to quickly and efficiently distribute large or critical content to huge populations of clients. When dealing with streaming or time-sensitive data, the content must be provided at a rate which is sufficient for its intended purpose (e.g., displaying a streaming movie).

In this paper, we have studied the problem of providing uniform download rates to a population of peers with asymmetric and heterogeneous bandwidth capacities. The architectures that best achieve this goal among those studied in the paper are the mesh and the parallel tree, but the latter requires peers to buffer data for a duration proportional to the depth of the trees. As the number of chunks grows, i.e., when the stream duration becomes very long, the differences between all the architectures become insignificant.

Although we only focused on analytical models for simple content distribution architectures, we believe that our analysis provides some important insights as how to set up peer-to-peer networks for distributing streaming data. It can also guide the design of cooperative applications that organize the nodes in a more dynamic manner than chains or trees. In particular, the system needs to build up upload capacity as fast as possible (which corresponds to maximizing the number of expansion steps) and the content should be partitioned into a large number of chunks (but not too many chunks as each one adds some coordination and connection overhead). By properly combining high and low capacity nodes, one can provide a high quality of service to every peer and even out their differences in a truly cooperative manner.

Acknowledgments. This work is supported in part by the Swiss National Foundation Grant 102819.

References

1. Chu, Y., Rao, S., Zhang, H.: A case for end system multicast. In: Proceedings of ACM Sigmetrics. (2000)
2. Hefeeda, M., Habib, A., Boyan, B., Xu, D., Bhargava, B.: PROMISE: peer-to-peer media streaming using CollectCast. Technical Report CS-TR 03-016, Purdue University (2003)
3. Jiang, X., Dong, Y., Xu, D., Bhargava, B.: Gnustream: A P2P media streaming system prototype. In: Proceedings of the International Conference on Multimedia and Expo (ICME),. (2003)
4. Chawathe, Y.: Scattercast: An adaptable broadcast distribution framework. Multimedia Systems **9** (2003) 104–118
5. Castro, M., Druschel, P., Kermarrec, A.M., Nandi, A., Rowstron, A., Singh, A.: SplitStream: High-bandwidth multicast in a cooperative environment. In: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP). (2003)
6. Jannotti, J., Gifford, D., Johnson, K.L., Kaashoek, M.F., O'Toole, J.W.: Overcast: Reliable multicasting with an overlay network. In: Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI). (2000)
7. Pendarakis, D., Shi, S., Verma, D., Waldvogel, M.: Almi: An application level multicast infrastructure. In: Proceedings of USITS. (2001)
8. Padmanabhan, V., Wang, H., Chou, P.: Resilient peer-to-peer streaming. In: Proceedings of IEEE ICNP. (2003)
9. Rao, S., Padmanabhan, V., Seshan, S., Zhang, H.: The impact of heterogeneous bandwidth constraints on dht-based multicast protocols. In: Proceedings of the 4th International Workshop on P2P Systems (IPTPS). (2005)

10. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scribe: a large-scale and decentralized application-level Multicast infrastructure. IEEE Journal on Selected Areas in Communications **20** (2003) 1489–1499
11. Biersack, E., Rodriguez, P., Felber, P.: Performance analysis of peer-to-peer networks for file distribution. In: Proceedings of the 5th International Workshop on Quality of future Internet Services (QofIS'04). (2004) 1–10
12. Yang, X., de Veciana, G.: Service capacity of peer-to-peer networks. In: Proceedings of IEEE INFOCOM. (2004)
13. Carra, D., Cigno, R.L., Biersack, E.: Introducing heterogeneity in performance analysis of p2p networks for file distribution. Technical Report DIT-04-113, University of Trento (2004)