Stochastic Graph Processes for Performance Evaluation of Content Delivery Applications in Overlay Networks

Damiano Carra, Renato Lo Cigno, *Member*, *IEEE*, and Ernst W. Biersack

Abstract—This paper proposes a new methodology to model the distribution of finite-size content to a group of users connected through an overlay network. Our methodology describes the distribution process as a constrained stochastic graph process (CSGP), where the constraints dictated by the content distribution protocol and the characteristics of the overlay network define the interaction among nodes. A CSGP is a semi-Markov process whose state is described by the graph itself. CSGPs offer a powerful description technique that can be exploited by Monte Carlo integration methods to compute in a very efficient way not only the mean but also the full distribution of metrics such as the file download times or the number of hops from the source to the receiving nodes. We model several distribution architectures based on trees and meshes as CSGPs and solve them numerically. We are able to study scenarios with a very large number of nodes, and we can precisely quantify the performance differences between the tree-based and mesh-based distribution architectures.

Index Terms—Modeling techniques, performance attributes, stochastic processes.

1 Introduction

The peer-to-peer (P2P) networking paradigm has received a lot of attention in recent years. P2P systems construct an overlay at the application layer and do not require any modification to the existing Internet, which is in contrast to other technologies such as IPv6 or IP-level multicast, which do require modifications inside the network. Therefore, P2P systems are very attractive for supporting new communications paradigms such as "application-level multicast" or "distributed publish/subscribe."

One of the most popular P2P applications is file sharing, a variation of which can also be used for file distribution. P2P for file distribution has the appealing feature of self-scaling: Each peer can play both roles, the one of a client and the one of a server, which implies that the amount of resources scales with the demand for service. The main difference between file sharing and file distribution is the different roles played by the search and download phases: In file sharing, the most difficult part is normally *finding* the file, whereas the download is considered just a matter of time. In file distribution, the file location is known, whereas the download phase is crucial, especially for very large numbers of downloading peers. The issue that needs to be addressed is how the resources of the overall P2P system can be used *efficiently*.

In this paper, we consider the specific problem of how we can distribute in the shortest possible time a file to a community of users organized as an overlay of peers. We develop a model that takes into account the transmission bandwidth between logically adjacent nodes as only "undetermined" of the problem, i.e., the only quantity that has a stochastic description. Extensions to other nondeterministic behaviors such as malfunctioning or malicious nodes are feasible: In Section 6, we present some results in the presence of nodes that leave the system. The computation of the file distribution time may seem simple. However, as discussed below, few results exist that go beyond the deterministic case where the bandwidth between peers is assumed to be constant and homogeneous.

We do not attempt to model a specific file distribution system, although this task may be possible by using the proposed methodology. We are also not proposing yet another file swarming application or a specific file distribution protocol. Rather, we propose a methodology to capture the general properties of distribution protocols and algorithms in an attempt to provide guidelines for the protocol and system design.

In particular, we provide a high-level description of a *generic* distribution protocol that can be implemented for delivering software patches or antivirus updates in static scenarios, such as enterprise or content replication to the edge, where congestion is not significant. In such a context, we can assume that users are collaborative. It is therefore not necessary to consider all the functionalities (for example, tit-for-tat) provided by well-known protocols such as BitTorrent. The aim of the content distribution application is to deliver the content in the fastest way and, thus, it is possible to rely on distribution architectures such as trees

- *D. Carra and R. Lo Cigno are with the Dipartimento di Informatica e Telecomunicazioni (DIT), Università di Trento, Via Sommarive 14, I-38050 POVO (TN), Trento, Italy. E-mail: {carra, locigno}@dit.unitn.it.*
- *E.W. Biersack is with the Corporate Communications Department, Institut EURECOM, 2229 route des cretes, F-06560 Sophia-Antipolis, France. E-mail: erbi@eurecom.fr.*

Manuscript received 25 July 2006; revised 12 Feb. 2007; accepted 10 Apr. 2007; published online 9 May 2007.
Recommended for acceptance by M. Ould-Khaoua.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0203-0706.
Digital Object Identifier no. 10.1109/TPDS.2007.1114.

or meshes built while distributing the content, which can be considered stable during the distribution process.

1.1 Prior Work

Performance analysis in terms of the minimum time required to distribute a file by using a P2P system has received some attention in recent years. Most of the analytical work focuses on a specific system and not on a generic distribution architecture. The work in [1] is among the first to evaluate the performance of a P2P system using a multiclass closed queuing network. In [2], the authors use an age-dependent branching process to model the transient evolution of a P2P system and a simple Markovian model to analyze the steady-state regime. Fluid models have recently been considered since they can efficiently describe the amount of transferred data. The work in [3] proposes a fluid model to accurately estimate the performance of the Squirrel protocol. In [4], the authors study the BitTorrent protocol [5] with a simple fluid model, which is able to catch the transient and the steady-state behavior of the system with few simple parameters. Moreover, an analysis of the different mechanisms of BitTorrent is provided. Among all these papers, only [1] tackles the problem of different access bandwidths among peers, which is also treated in this paper.

A related topic where distribution architectures are explicitly taken into account is the delivery of streaming services through overlay multicast. Application-Level Multicast Infrastructure (ALMI) [6] and SplitStream [7] define a set of mechanisms to efficiently distribute the streaming data to many overlay nodes. They build distribution trees in different ways and manage the dynamics of nodes that are leaving or joining. Nevertheless, most of these studies are focused on the protocol design and do not quantify the impact of the distribution architecture on performance. The performance evaluation is limited to specific aspects of the proposed protocol and not on the entire network.

To the best of our knowledge, very few models that allow comparative studies of different distribution architectures have been proposed. In [8], inspired by SplitStream, the authors have defined and analyzed linear-chain-based and tree-based architectures in a completely deterministic setting. The work in [9] defines a model for chain-based and tree-based architectures, uses max-plus algebra, and considers an infinite number of packets to calculate the long-term average throughput. Our analysis instead considers a finite file size and calculates either the download time of peer i or the mean (and total) download time of all the peers involved in the distribution process.

Stochastic Graph Processes (SGPs), which are the analytical tool that we use in this paper to model overlay content delivery networks, were defined in [10], with the same notation that we use here, and they are based on well-known random graphs [11]. The focus of the analysis in [10] and [11] is the topological properties of random graphs, whereas our aim is to take into account not only connections among nodes but also their weights (lengths) given by the bandwidths of the nodes involved (this concept is clarified in Section 3.1), which give rise to the state reward structure that allows the computation of completion times. Moreover, content distribution is done by building a content distribution graph *on top* of the overlay graph (see Fig. 1), and these two levels have different properties.

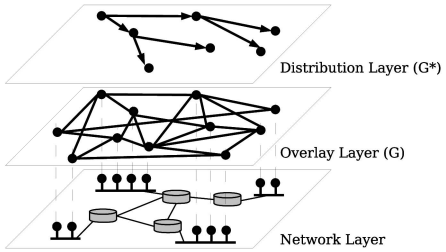

Fig. 1. Overlay and distribution graphs.

1.2 Contributions of This Paper

In Sections 3 and 4, we formalize the problem of building a content distribution overlay as a Constrained Stochastic Graph Process (CSGP), which is a discrete-time Markov chain (DTMC) whose states are graphs with additional constraints describing the features of the distribution system. The graphs that we are interested in are directed acyclic graphs such as trees or meshes. Depending on how we define the transition rules from one state of CSGP to the next one, we get different content distribution overlays. We consider three different overlays, two of which are trees (hop-driven and cost-driven trees), and the third of which is a mesh. Part of the material (related to tree architectures) of these initial sections was presented in [12], where the initial ideas about the use of CSGP as modeling techniques for complex systems were discussed.

In Section 5, we explain in detail the solution approach and the motivations underlying the use of the proposed methodology. We show how we can use the formalism of CSGP to compute the metrics of interest for the content distribution, such as the file download times and the percentage of upload bandwidth left unused. Numerical results are obtained by solving the stochastic process via Monte Carlo integration. Monte Carlo integration converges very quickly, allowing us to obtain results for very large overlays with up to a million nodes, where standard event-driven simulations would fail for the lack of time or memory. Monte Carlo integration has been used in other fields such as physics and chemistry, but to the best of our knowledge, it has never been applied to modeling overlay networks. The interested reader is referred to [13], where we briefly discuss the similarities between modeling the content propagation across an overlay and the modeling of chemical reactions of a set of molecules.

In Section 6, we present the results for the three different distribution overlays. These results precisely quantify, for the first time, the impact of the minimum outdegree and the improvement of mesh-based overlays, as compared to tree-based overlays, and provide important practical insights into how content distribution overlays should be constructed. Some of these results were presented in a

preliminary form in [14]. In Section 8, we discuss practical issues for the design of content distribution algorithms.

2 Overlay Content Delivery Applications

Content Delivery Applications are systems where information (data or multimedia) is distributed to a community of users. They include fast delivery of antivirus updates, software distribution, live video streaming, or video-on-demand distribution. Different types of contents have different requirements (for example, maximum delay), but they can share common features and fundamental architectures.

The content is distributed by using an overlay network formed by a P2P application connecting end hosts. In this paper, we focus on the distribution of a finite-size file to cooperative users willing to forward the content to others. We consider a BitTorrent-like distribution protocol where the file is broken into independent pieces called *chunks*. A node that has started to download the file can, in turn, start to upload after it has entirely received the first chunk. The order in which the chunks are received by a node is not important. However, the transfer is not complete until all the chunks have been received by all interested nodes.

The distribution application has different configuration parameters. The source of the content can decide the size of the chunks (or, alternatively, the number of chunks). A typical chunk size is 256 Kbytes: The download time for a single chunk is typically much larger than the propagation delay. Control messages represent few packets with information about the chunks received and the traffic generated can be considered negligible with respect to the chunk transmission. Users can also set the maximum download and upload bandwidth, as well as the maximum number of active connections used for data exchange.

The path followed by chunks is composed of a subset of edges of the overlay network. This subset, which is determined by the configuration parameters of the application (for example, the maximum number of upload connections), identifies the *distribution architecture*: The distribution architecture will have a major impact on the time that it takes to distribute the content. Thus, one of the major issues in content distribution networks is the choice of the distribution architecture.

Although we consider the P2P overlay network already formed, the distribution graph is built step by step while distributing the content; that is, edges are included when nodes start downloading the file.

3 Problem Formulation

In this section, we formalize the problem of delivering a given finite-size content $\mathcal{F}$ to a set of users $\mathcal{N}$. We propose a novel methodology to model and analyze file distribution that is based on a class of semi-Markov processes whose state is described through a graph. This property allows adding a reward structure to the process that is related to the topological properties of the graph, which enables the computation of the performance metrics and gives deep insight into the behavior of the file distribution systems.

3.1 Basic Assumptions

We consider a basic distribution protocol, where the aim is to deliver the content to cooperative users: We do not consider advanced features such as tit-for-tat implemented in BitTorrent, focusing on basic parameters that influence the distribution protocol.

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

The content $\mathcal{F}$ is divided in C pieces. We assume that each node knows a subset of the whole set of users; that is, a node has a finite number of neighbors.

We initially consider that nodes are stable; that is, they always stay online. We then relax this assumption (details on the node leaving process can be found in Section 6.1), showing the impact on the performance metrics. For each node i, we define b_i^u and b_i^d as the upload and download bandwidth, respectively, which can be symmetric, asymmetric, or correlated, e.g., $b_i^u + b_i^d$ constant, as in a shared-medium-based access. The values of the bandwidth in the network are described through a probability density function (pdf) that is known (e.g., derived from measurement studies). We assume that the upload/download bandwidth remains constant during the distribution process: Since the download time can be in the order of hours, we consider the bandwidth as a mean value as fluctuations are much faster than the download time.

When a node starts uploading chunks of $\mathcal{F}$, the *effective rate* that is used to transfer the chunks to each child depends on multiple factors such as the number of children of the uploading node and the rate at which the uploading node is receiving chunks. In particular, given the upload bandwidth and the upload rate[1] of the parent node i, b_i^u and r_i, and the download bandwidths of the children nodes,[2] $b_j^d \; \forall j \in \{\text{children set}\}$, the single chunk transfer rate to each child j, b_j^{*d}, is computed according to the *max-min fairness* criterion [15]. The effective rate is then calculated as the minimum between the single chunk transfer rate and the effective rate of the parent node $r_{ij} = \min(b_j^{*d}, r_i)$. While the upload and download bandwidths are given, the effective rates are computed during the distribution process.

We define the *eligibility time* t_i^{el} of node i as the time at which node i can start uploading chunks to other nodes; i.e., it has completely received the first chunk. We assume that each node receives the *first chunk* from a *single parent*. Once it has completely downloaded the first chunk, it can accept other incoming connections. This assumption simplifies the calculations and has only a negligible effect if the number of chunks C is much greater than one. If a node j is child of node i and receives at an effective rate r_{ij}, its eligibility time is $t_j^{\text{el}} = t_i^{\text{el}} + t_{ij}^{\text{step}}$, where $t_{ij}^{\text{step}} \triangleq \frac{\mathcal{F}}{C}\frac{1}{r_{ij}}$. In case of tree-based distribution architectures, where each node has a single parent, knowing the eligibility time and the rate at which a node j receives chunks, the *total download time* can be computed as $T_j \triangleq t_j^{\text{el}} + (C-1)t_{ij}^{\text{step}}$ (at the end of the

1. The upload rate is the minimum between the receiving rate of the node and its upload bandwidth.

2. How many children are selected by the parent is decided according to a *saturation* criterion defined in Section 4.1.

eligibility time, the first chunk is received, so there are $C-1$ chunks left). In case of mesh-based architectures, the download time is computed by considering the rates of all parents and the instants when the different chunks are received.

As a last assumption, we suppose that node i chooses its children j uniformly at random among all its neighbors, not taking into account the upload and download bandwidths.

3.2 General Definitions

The distribution of content within a community of users can be formalized as the propagation of the content across a graph of nodes and edges with some stochastically defined characteristics. Nodes are the users and edges summarize all the characteristics of the communication paths between the users.

Let $\mathcal{N}$ be the set of nodes, i.e., the vertices of the graph, and $\mathcal{A}$ be the set of all the arcs that connect pairs of nodes $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. We only consider connected networks with bidirectional connections so that $\mathcal{A}$ can be represented by an irreducible symmetric adjacency matrix. $\mathcal{B}_i$ is the set of neighbors of user i, i.e., all those nodes in $\mathcal{N}$ that are known and directly reachable from node i, with $\bigcup_{i\in\mathcal{N}} \mathcal{B}_i = \mathcal{N}$, since the network is fully reachable. $\mathcal{B}_i$ is also represented by row i of the adjacency matrix $\mathcal{A}$.

The graph $\mathcal{G}(\mathcal{N},\mathcal{A})$ represents the overlay network created, for instance, by a P2P network (see Fig. 1). For each node i, the neighbor set $\mathcal{B}_i$ is the subgraph of $\mathcal{G}$ containing the neighbors of i and the related arcs. In general, $\mathcal{G}(\mathcal{N},\mathcal{A})$ is time varying; i.e., nodes and edges can change in time and even appear or disappear. The overlay layer is the basis on top of which the *distribution graph* is built. We define the distribution graph $\mathcal{G}^*(\mathcal{N},\mathcal{E})$ as a directed subgraph of $\mathcal{G}(\mathcal{N},\mathcal{A})$ with $\mathcal{E} \subseteq \mathcal{A}$. $\mathcal{G}^*$ is a directed graph, since, from the content distribution point of view, the content propagates from the source to the destinations.

3.3 Stochastic Graph Processes for Content Distribution

How we can obtain the distribution graph $\mathcal{G}^*$ from $\mathcal{G}$ is determined by the rules implemented in the specific content distribution protocol. In general, we can assume that the distribution graph $\mathcal{G}^*$ is built step by step following a given protocol. The building process can be modeled as a DTMC since the distribution graph $\mathcal{G}^*$ at step n contains all the information required to (stochastically) define the distribution graph $\mathcal{G}^*$ at step $n+1$. Let $\mathcal{N}_n^*$ be the set of nodes that belong to the distribution graph at step n and let $\overline{\mathcal{N}_n^*}$ be its complement with respect to $\mathcal{N}$. The distribution graph $\mathcal{G}_{n+1}^*$ at step $n+1$ is obtained from $\mathcal{G}_n^*$ by adding new edges $\in \mathcal{A}$ from nodes in $\mathcal{N}_n^*$ to nodes in $\overline{\mathcal{N}_n^*}$. The complete distribution graph $\mathcal{G}^*(\mathcal{N},\mathcal{E})$ is obtained when $\mathcal{N}_n^* = \mathcal{N}$, and $\overline{\mathcal{N}_n^*}$ is the empty set.

The dynamic behavior of the distribution graph can be modeled as an *SGP*. We recall here the general definition of SGPs [10], whereas, in Section 4, we specialize them for the analysis of content distribution.

Definition 1. *An SGP on a set of nodes $\mathcal{N}$ is a DTMC whose states are graphs on $\mathcal{N}$.*

Even if not stated in the definition, two observations are in order:

- Nodes can be connected only through edges that belong to $\mathcal{A}$ (in the next definitions, we state explicitly this dependence).
- The SGP is embedded in a continuous time semi-Markov chain that is sampled at the instants of adding nodes and arcs to obtain the SGP.

Adhering to the definition given in [10], the focus is on the building process, and the SGP evolution implies that the graph is built step by step by adding nodes and edges at each step.

In content distribution, the distribution graph is naturally built step by step, so using the graph $\mathcal{G}^*(\mathcal{N},\mathcal{E})$ as a formal representation of the state of the system is appropriate and complete; that is, that state contains all the information required to define (stochastically) the next state of the evolution. The time between two steps depends on the sojourn time of the state. If the sojourn times are exponentially distributed, then we obtain a Markov chain. However, in general, this assumption is not true, and in continuous time, we have a semi-Markov chain.

Definition 2. *A CSGP on a graph $\mathcal{G}(\mathcal{N},\mathcal{A})$ is a semi-Markov chain whose states are subgraphs on $\mathcal{G}$. The semi-Markov chain embeds a DTMC obtained by sampling the process exactly at transition instants. The "constraints" limit the degrees of freedom during state transitions and "govern" the evolution of the process.*

A CSGP is a precise representation of the content distribution process.

Given $\mathcal{G}_n^*$, the next state $\mathcal{G}_{n+1}^*$ depends only on the eligibility times of the nodes in $\mathcal{N}_n^*$, and the transition probabilities can be easily defined step by step.

The eligibility times t_j^{el} influence the semi-Markov process in two different ways. In the general case of randomly varying t_{ij}^{step}, they define both the transition probabilities between states and the state sojourn times. In the particular case of deterministic t_{ij}^{step} (e.g., when the bandwidth is only determined by access links), the sojourn times are deterministic and the t_j^{el} define only the state-transition probabilities. Notice, however, that the file distribution is entirely described by the embedded DTMC so that only transition probabilities are important.

The DTMC that describes a CSGP is a transient chain with a set of adsorbing states $\mathcal{G}^*(\mathcal{N},\mathcal{E})$ that are reached when $\mathcal{N}_n^* = \mathcal{N}$; i.e., all nodes interested in the content are reached.

The way that we defined a CSGP implies that nodes are stable and collaborative and that the networking infrastructure is reliable enough to allow edge stability. Clearly, there is the possibility of extending the analysis to cases where nodes (or edges) can disappear during the distribution process so that $\mathcal{G}_n^*$ is derived from $\mathcal{G}_{n-1}^*$ not only by adding an edge and a node but also by removing one node and all the edges relative to it.

A well-known class of CSGPs are the *random graph processes* studied back in the 1950s by Erdös and Renyi [11]. In random graph processes, edges are added uniformly at random, which does not capture the propagation of the data blocks in content distribution graph.

4 Content Delivery CSGP

In the following, we refine the CSGP by adding constraints. These constraints concern the minimal and maximal outdegree and also the depth of graph. We define three different distribution architectures, referred to as Content-Delivery Constrained Stochastic Graph Processes (CD-CSGPs). Two of the CD-CSGPs form trees, and the third one forms a mesh. We have seen that the distribution graph, as defined by a CD-CSGP, grows step by step by connecting each time a new node to a node i in the existing graph. As we will see for the two tree-based distribution architectures, the final distribution tree will be very much affected by the way that we select the node i in the graph to which the new node will be attached. The mesh-based distribution graph will be obtained by first constructing a tree-based distribution graph, which will then be augmented by additional links originating at the leaves of the tree. The advantage of a mesh is that the leaf nodes of a tree, which do not at all contribute to the distribution of the content, will now help distribute the file content.

4.1 Content-Delivery-Related Definitions

Before we introduce the CD-CSGP, we need some additional definitions that will simplify the characterization of each CD-CSGP.

Each node has a constraint on the maximum and minimum numbers of active uploads that limit the possible outdegree of the node: $k_i^{\max}$ is the *maximum outdegree* and $k_i^{\min}$ is the *minimum outdegree*.

Definition 3 (saturated node). *A node $i \in \mathcal{N}_n^*$ is called* saturated *if it has either*

- *$k_i^{\max}$ outgoing edges that belong to $\mathcal{G}_n^*$ or*
- *at least $k_i^{\min}$ outgoing edges, and the sum of the download rates to its children is equal to b_i^u.*

Definition 4 (interior subset). *The subset $\mathcal{I}_n \subseteq \mathcal{N}_n^*$ of nodes that are saturated at step n is called the interior node subset at step n.*

Definition 5 (leaf subset). *The set of nodes $\mathcal{L}_n \in \mathcal{N}_n^*$ that are not interior nodes is called the leaf node subset at step n, with $\mathcal{L}_n = \mathcal{N}_n^* \setminus \mathcal{I}_n$.*

We consider a single node as a root of the stochastic graph. We define a distance measure based on the number of hops from the root to any node i.

Definition 6 (step distance). *The number of hops from the root to a node i following the shortest path is called* step distance *or* step depth $d^{(i)}$.

In a tree, $\max_i(d^{(i)})$ is the *tree depth*.

4.2 Cost-Driven and Hop-Driven Trees

We now define the precise steps that must be executed in building the content distribution tree. The exact rules for building the tree have a big impact on the shape (as well as number of hops and outdegree) of the final content distribution tree. In the following, we will introduce two different trees called *cost-driven* and *hop-driven* trees.

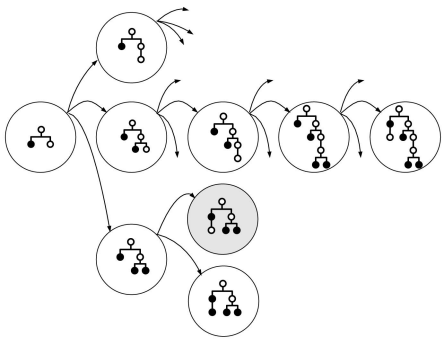

Fig. 2. Sample of the embedded DTMC for a CD-CSGP 1 process. States are graphs built on $\mathcal{G}$, black and white circles represent slow and fast nodes, respectively, $k_i^{\max} = 2$ and $k_i^{\min} = 1$.

CD-CSGP 1. A CSGP on graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$ is called **tree-based** and **cost-driven** if the following hold true:

1. $\mathcal{G}_0^*$ is a node, called root, randomly chosen in $\mathcal{N}$.
2. $\mathcal{G}_n^*$ is obtained from $\mathcal{G}_{n-1}^*$ as follows:
   - Choose the node i from $\mathcal{L}_{n-1}$ with the smallest eligibility time $t_i^{\text{el}} = \min_j(t_j^{\text{el}})$. If several nodes have the same eligibility time, one of the nodes is chosen randomly.
   - Add edges from node i to nodes randomly chosen from $\mathcal{B}_i \bigcap \overline{\mathcal{N}_{n-1}^*}$ until node i becomes saturated.

Fig. 2 shows an example, with few states of the DTMC generated by a CD-CSGP 1 process. In this case, we have only two possible bandwidths (slow nodes with black circles and fast nodes with white ones, with the slow bandwidth less than half the fast bandwidth), $k_i^{\max} = 2$ and $k_i^{\min} = 1$. Starting, for instance, from a state where the server is uploading to a slow node and to a fast node, the fast node has the smallest eligibility time and there are only three next possible states:

1. The fast node selects a fast node among its neighbors and becomes saturated. Alternatively, the fast node chooses a slow node, so it has to select another node.
2. The selected node is fast and we have bandwidth saturation.
3. The node chosen is slow and we have saturation because $k^{\max}$ is reached.

Note that, in case 1, the node becomes saturated since the rate of the content that it is receiving is high. If, for instance, the rate were slow (consider the fast node under the slow node in the shadowed state), the number of children would be always 2 since the rate to each child is at most equal to the rate that it is receiving.

The resulting tree is called "*cost driven*" since, in general, the nodes in the leaf set $\mathcal{L}_n$ do not all have the same step distance from the root. As the speed of growth of the different branches is not the same, the deeper branches will contain faster nodes, that is, nodes with smaller eligibility times t^{el} (eligibility times represent the costs).

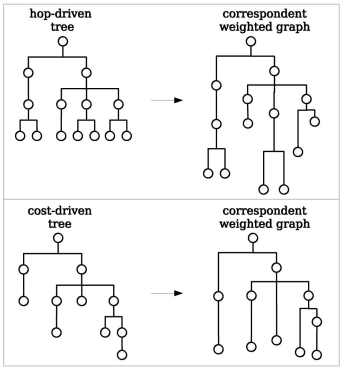

Fig. 3. Difference between hop-driven and cost-driven trees, considering the corresponding weighted graphs, where the length of edge between nodes i and j is given by t_{ij}^{step}.

Many analytical models proposed in the literature consider balanced trees, for example, binary trees. This assumption is actually unrealistic. Forcing the same step distance for all the leaves means that we do not consider the bandwidth heterogeneity when chunks are transferred. Nevertheless, the majority of the proposals *consider* trees where leaves are at the same step distance from the source, even in cases of variable outdegree. We call such trees "*hop driven*." These trees do not represent real distribution applications, but we have to define a special SGP for them in order to reproduce results that can be found in the literature and compare these results with those obtained with more realistic architectures such as cost-driven trees. To do so, we consider a subset of the leaf set $\mathcal{L}_n$.

Definition 7. *Let* $d_n^{\text{MAX}} = \max_j(d_n^{(j)})$ *be the maximum step distance of the nodes* $j \in \mathcal{L}_n$. *The subset* $\widetilde{\mathcal{L}_n} \subseteq \mathcal{L}_n$ *is defined as* $\widetilde{\mathcal{L}_n} = \{i \in \mathcal{L}_n | d_n^{(i)} < d_n^{\text{MAX}}\}$.

Now, we can define the process that leads to hop-driven trees.

CD-CSGP 2. A CSGP on graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$ is called **tree-based** and **hop-driven** if the following hold true:

1. $\mathcal{G}_0^*$ is a node, called root, randomly chosen in $\mathcal{N}$.
2. $\mathcal{G}_n^*$ is obtained from $\mathcal{G}_{n-1}^*$ as follows:
   - Choose a node i from $\widetilde{\mathcal{L}_{n-1}}$ if it is not empty; otherwise, choose from $\mathcal{L}_{n-1}$ with the smallest eligibility time. If several nodes have the eligibility time, one of the nodes is chosen randomly.
   - Add edges from node i to nodes randomly chosen from $\mathcal{B}_i \bigcap \overline{\mathcal{N}_{n-1}^*}$ until the node becomes saturated.

Since we are interested in the file download time, it is worth looking at a weighted graph where the weight associated to a directed edge is given by the difference between the download times of the nodes connected by the edge. Considering hop-driven trees, this representation shows the disparity in terms of download time among leaf nodes that have the same step distance. In Fig. 3, the weight is represented by the edge length. Conversely, in cost-driven trees, leaf nodes are at different step distances and the weighted graph gives a pictorial illustration of why the tree grows this way: A new edge is added only after a node becomes eligible and this forces a uniform growth of the *weighted* graph.

The difference between hop-driven and cost-driven trees is significant. A node i in the tree will influence the reception speed of all nodes in the subtree with i being the root. A slow node j will slow down the reception of the chucks for all the nodes in the subtree with j being the root. Since slow nodes become eligible later than fast nodes, in the case of cost-driven tree, a subtree with a slow node being its root will grow much slower, i.e., have fewer nodes than a subtree consisting of fast nodes, which will help "limit" the impact of slow nodes.

We will show in Section 6 how the choice of the tree, hop-driven or cost-driven, affects the performance.

4.3 General Mesh Architecture

Tree-based architectures allow the content to rapidly diffuse to nodes. However, trees also have known shortcomings. Each node has only one ancestor, and in case of a node failure, the entire subtree will stop receiving data. Each node must divide the upload bandwidth among its children, so children use only a fraction of their download bandwidth for receiving chunks. If we consider the case of asymmetric capacities, where the upload bandwidth is smaller than the download bandwidth (as in the case of the Asymmetric Digital Subscriber Line (ADSL)), the percentage of unused download bandwidth increases even further. Finally, the leaf nodes of a tree receive the entire file without uploading a single chunk.

Mesh-based architectures are meant to overcome these problems. Nevertheless, they introduce a new problem: the chunk selection strategy. A node i can help another node j if i has parts of $\mathcal{F}$ that are not yet received by j, i.e., if it has "fresh" information. We are not concerned here on how freshness is checked and/or imposed (for instance, see [16] for work on the topic). We assume an ideal situation, where, if a node has received only part of $\mathcal{F}$ and it is contacted by another node that is not yet its ancestor, then all the information that it can provide is either completely fresh or completely stale. Any impairment can be easily taken into account with a probabilistic approach.

Allowing the generation of mesh topologies means that a node i already included in $\mathcal{G}_n^*$ may be contacted by other nodes j that are also in $\mathcal{G}_n^*$ to receive parts of $\mathcal{F}$ that it does not yet have. If j has only information that is not fresh for i, then the "delivery connection" is not established.

The building process of a mesh architecture can be divided into two phases: diffusion and interconnection. In the first phase, we assume that the root node uploads the chunks to its k children in different orders. For instance, child i will first receive the chunk j, with $j \in \{1, \ldots, C\}$, and $j \bmod k = i$. This means that each child receives the whole file, but the first C/k chunks are disjoint with respect to the content received by the other children.[3] Each of the children generates its own *diffusion subtree* $\mathcal{FG}$, where new nodes that

3. Other techniques such as network coding could also be used to assure that different children receive different chunks, but these details do not impact the performance of distribution architectures.

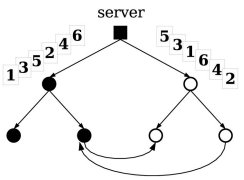

Fig. 4. Mesh topologies obtained from the interconnection of different diffusion subtrees.

are not yet reached by the content (*untouched nodes*) are added to the subtrees. Once no more untouched nodes are available, subtrees start to interconnect. Leaf nodes of a subtree upload chunks to nodes belonging to different diffusion subtrees, providing up to C/k disjoint chunks. In the final configuration, we obtain a mesh, where each node receives from up to k nodes (belonging to distinct diffusion subtrees) and uploads to other nodes according to the saturation rules. This constrained mesh diffusion process is depicted in Fig. 4. Note that the diffusion direction is "reverted" at the leaves, ensuring a better spreading of the content.[4]

In order to improve the performance, we require that diffusion trees must grow until no more untouched nodes are available. In fact, [8] shows that, in the homogeneous case, the best performance can be obtained if the node is an internal node of one diffusion tree and a leaf node in the remaining trees. Untouched nodes have some upload bandwidth available, so they can become internal nodes. Nodes that have already received the content probably have already started to distribute it and have little upload bandwidth available, slowing down the growth of the diffusion tree. This policy can be implemented in a distributed manner: When a node finishes downloading the first chunk, it first searches for untouched neighbors, and if it is not able to find any, it starts uploading to nodes that are already downloading.

We can formally define the CSGP that leads to these architectures.

CD-CSGP 3. A CSGP on graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$ is called constrained **mesh-based** if the following hold true:

1. $\mathcal{G}_0^*$ is a node, called root, randomly chosen in $\mathcal{N}$.
2. $\mathcal{G}_1^*$ consists of the root node and the k nodes randomly chosen in $\mathcal{N} \setminus \{root\}$ called the first generation nodes, where each of them will generate a subtree $\mathcal{FG}$.
3. $\mathcal{G}_n^*$ is obtained from $\mathcal{G}_{n-1}^*$ as follows:
   - Choose a node i from $\mathcal{L}_{n-1}$ with the smallest eligibility time. If several nodes have the same eligibility time, one of the nodes is chosen randomly. $\mathcal{FG}_i$ denotes the subtree to which i belongs.
   - Add edges from node i to nodes randomly chosen from $\mathcal{B}_i \bigcap \overline{\mathcal{N}_{n-1}^*}$ until the node becomes saturated and $\overline{\mathcal{N}_{n-1}^*}$ is not empty.
   - If $\mathcal{B}_i \bigcap \overline{\mathcal{N}_{n-1}^*}$ is empty and node i is not saturated, add edges from node i to nodes randomly chosen from $\mathcal{B}_i \bigcap \mathcal{N}_{n-1}^{\dagger}$ until the node becomes saturated, where $\mathcal{N}_{n-1}^{\dagger}$ is the set of nodes in the subtree $\mathcal{FG}_i$ at step $n-1$.

CD-CSGP 3 can describe construction processes such as *SplitStream* [7] and *PTree* [8]: Both schemes use a fixed outdegree k and partition the file in exactly k stripes, where each stripe is distributed along one of the k diffusion trees. The process that we define is more general since we do not impose any fixed outdegree and allow that nodes upload the whole file, however, in different orders.

5 Solution of the CD-CSGP

The CD-CSGPs defined in Section 4 describe the evolution over time of the underlying Markov process: The Markov chain that models the process is a transient chain and its absorbing states are the states where all elements of $\mathcal{N}$ have downloaded $\mathcal{F}$. Thus, we are interested in the analysis of the transient behavior of the system; i.e., we focus on the time that is necessary to reach an absorbing state or, given a time bound, the mean number of nodes contained in the distribution graph.

The transient analysis of a Markov process can be done by considering the well-known Kolmogorov forward (or backward) equations, coupled with the Chapman-Kolmogorov equations, which lead to a set of differential equations that describe how the probabilities to be in a given state change over time. A closed-form solution of these equations is not feasible unless it is for a very small number of nodes, so we need to resort to numerical integration.

A numerical solution of the Markov process equations can be done by using different methodologies. Direct methods that numerically approximate the equations have similar problems as the closed form solution: The equations can be written only for a small number of nodes.

However, the structure of the transition matrix that describes the stochastic process is extremely suited for an efficient numerical solution based on Monte Carlo techniques [17], [18].[5]

Monte Carlo integration is basically a random walk in the state space of the process. The convenience of the methodology is given by the fact that it is very simple to build a random walk by following the process definitions given in Section 4. Each random walk starts from the initial empty state and finishes in an absorbing state. At each step, we generate transition probabilities at runtime when hitting a state.

These samples are, by construction, independent and identically distributed, so we can compute the average characteristics of these random walks, along with the confidence interval, given a desired confidence level.

4. It may be possible that a node has two parents, let us say, A and B, and uploads to parent A the content received from parent B, provided that A is not receiving from other nodes belonging to the diffusion subtree of B.

5. In physical and chemical sciences, this technique is often called the *Stochastic Simulation Algorithm* or *Gillespie Algorithm*, but we prefer to stick to the term "Monte Carlo," which is normally used in computer science.

The Monte Carlo integration is efficient for a number of reasons:

- There are a small number of transitions from each state, all of them with a nonvanishing probability, so that rare paths are nonexisting.
- The reward structure defined on the DTMC (see Section 5.2 for the definition of the metrics that we use) ensures that there are no dominating rewards associated to low-probability states (no "rare-event" syndrome is present in the problem) and that the coefficient of variation (standard deviation divided by the average value) of all output metrics decreases as the number of nodes increases. In fact, the variance remains bounded, whereas the mean download time increases monotonically as we add more nodes. The consequence is that the numerical results yielding the histograms of the distribution converges (almost surely) more rapidly to the exact distribution as the number of nodes increases.
- The difference between the estimates and the exact value can be evaluated with standard stochastic means [19], yielding a powerful tool to stop the solution iteration.

The Monte Carlo integration that we propose is not a "generic" event-driven simulation but a numerical solution technique for the analytical model. The definition of the problem as a CSGP ensures that the numerical solution generates all and only the states that are relevant to compute the performance metrics. A generic event-driven simulator, written without formally defining the underlying stochastic process, may end up in exploring a state space where interesting states are sparse in the middle of states that are not useful to find the metrics of interest.

5.1 Detailed Description

The results of the Monte Carlo integration are an approximated solution of the differential equations that describe the process, but with a known error. All the probabilities that we compute are estimations of the real probabilities. For instance, when we find the performance metrics, as described in Section 5.2, we consider the probabilities of the absorbing states. Actually, these probabilities are the estimated probabilities with an error bound given by confidence intervals.

The fast convergence of the integration using Monte Carlo is due to the fact that we look for node properties that are not necessarily related to the graph structure. For instance, given a time t and a specific number of nodes N_t that have completed the download, there are many different possible graphs that contain N_t nodes at time t. Another example can be the download rates of nodes in the absorbing states: There are many graph structures where the number of nodes that complete the download with the slowest rate is the same. Thanks to these aggregate measures, the convergence of the distribution is fast.

5.2 Computation of the Performance Metrics

In order to compute the mean download time $\overline{T}$, we assign to each absorbing state $\boldsymbol{S}_k \in \boldsymbol{S}^a$ ($\boldsymbol{S}^a$ is the set of absorbing states; i.e., the states where all elements of $\mathcal{N}$ have downloaded $\mathcal{F}$ and no further transitions are possible) a reward $\overline{T}_k$ that is equal to the mean download time of the nodes in the state:

$$\overline{T}_k = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} T_i \ ; \ \boldsymbol{S}_k \in \boldsymbol{S}^a,$$

where T_i are the individual download times defined in Section 3.1. The mean download time $\overline{T}$ is the reward of a DTMC obtained by adding a deterministic transition from all the absorbing states to an initial state represented by the empty graph $\emptyset$:

$$\overline{T} = \frac{\sum_{k\in\boldsymbol{S}^a} \overline{T}_k \pi_k}{\sum_{k\in\boldsymbol{S}^a} \pi_k},$$

where π_ks are defined as the steady-state probabilities of the support DTMC.

Another performance measure that is easily defined as a reward is the *wasted upload bandwidth* $\overline{w^u}$ (in percentage). Let $w_i^u = 100\left(1 - \frac{\max(r_i^u)}{b_i^u}\right)$ be the wasted upload bandwidth of node i, where $\max(r_i^u)$ is the maximum upload rate ever reached by the node in any visited state. Considering again the modified DTMC and letting $\mathbf{S}_a$ be the set of absorbing states in the unmodified DTMC, we have

$$\overline{w_k^u} = \frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} w_i^u \ ; \ \mathbf{S}_k \in \mathbf{S}_a$$

and

$$\overline{w^u} = \frac{\sum_{k\in\mathbf{S}_a} \overline{w_k^u}\pi_k}{\sum_{k\in\mathbf{S}_a} \pi_k}.$$

If we fix the number of nodes and let the time to go to infinity, we obtain the pdf of the download rates and the number of nodes that download at each possible rate normalized by the total number of nodes. This representation enables the comparison of the input pdf of the bandwidths with the output pdf of the rates, obtaining a direct measure of the impact of different policies (for example, constraints on the outdegree) on the performance.

5.3 The Numerical Solver

We developed a numerical solver called stochastic Graph pROcess sOlVER (GROOVER), which implements a Monte Carlo integration in the form of an algorithmic implementation of the content distribution processes defined as CD-CSGPs. It simulates the stochastic process and computes multiple realizations of that process. GROOVER has several input parameters and produces as outputs the rewards associated with the process.

Algorithm 1 gives a high-level view of GROOVER for the case of a tree-based architecture, as defined in CD-CSGP 1. The computation of rates r_{ij} is done as explained in Section 3.1. A detailed description of GROOVER can be found in [14]. The software is available at [22].

Algorithm 1: Basic structure of GROOVER for tree-based architectures.

input: pdf of node bandwidth, total number of nodes, desired confidence level
output: pdf of download times;
initialization;
repeat
  while (#nodes $<$ tot.nodes) **do** /*build (diffusion) trees*/
    extract the node i with $(t_i^{\mathrm{el}} = \min_j(t_j^{\mathrm{el}}))$ **and** (#levels $<$ max level);
    repeat
      select randomly a neighbor from the $\mathcal{B}_i$ not yet reached by any other node;
    until node i is saturated
    if $(\#\mathrm{children} = 0)$ **or** (node not saturated) **then**
      put node in leaf set
    else
      compute the rate r_k to each children according to the *max min fairness criterion*;
      for $k = 1$ to #children **do**
        assign t_k^{el};
        compute $t_i^{\mathrm{download}} = t_k^{\mathrm{el}} + \frac{\mathcal{F}}{C}(C-1)/r_k$;
      end for
    end if
  end while
  update histograms (time, wasted bandwidth, etc.);
until stop criterion not met

stop criterion:
update the confidence interval for the histogram including the last realization; stop if desired confidence level is reached;

Solution Complexity. We have not attempted to compute a priori the number of necessary realizations to obtain the given confidence level and interval, although it may be possible to exploit the properties of the DTMC. We use instead an a posteriori evaluation of the distribution's accuracy by using standard methods (see [19]). As expected, the number of realizations needed to reach a desired accuracy decreases with the number of nodes.

For instance, in the numerical examples presented in Section 6, the convergence is obtained in less than 1,000 realizations for 10^4 nodes, less than 500 realizations for 10^5, and less than 200 realizations for 10^6. For 10^6 nodes and a mesh-based architecture, this means 4-5 hours of CPU on a standard PC and 10-20 minutes for 10^5 nodes, whereas, for a smaller number of nodes, the execution time becomes negligible.

For tree-based architectures, we also implemented a much faster algorithm that exploits the properties of the paths within the tree: We only realize a single path from the source to a leaf within the tree and use the properties of the DTMC to derive the metrics for the entire tree directly in the form of aggregate histograms. For more details on this method, see [14]. This further reduces the solution time by more than one order of magnitude, allowing the evaluation of tree architectures for up to 10^8 nodes. To the best of our knowledge, numerical results for overlay-based content distribution networks rarely extend beyond 10^3-10^4 nodes.

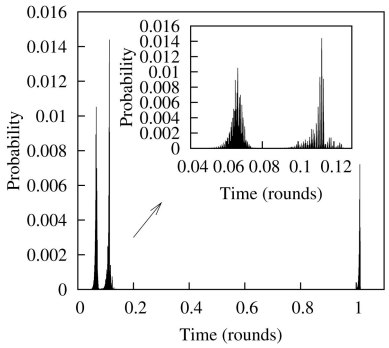

Fig. 5. Histogram of the estimated pdf of the download time of the nodes with CD-CSGP 3 for 10^4 nodes.

6 Numerical Results

To obtain our results, we use the pdf for the node bandwidth taken from [20] as input: 13 percent of the nodes with 56 kilobits per second (Kbps), 23 percent with 640 Kbps, and 64 percent of the nodes with 1.2 megabits per second (Mbps).

When reporting results, we normalize the data such that $\frac{|\mathcal{F}|}{\min_i(b_i)} = 1$ "round," where $|\mathcal{F}|$ is the content size in bits and $\min_i(b_i)$ is the minimum bandwidth of the input pdf in bits/s. We use a number of chunks C equal to 100, but a sensitivity analysis with different values of C indicates a qualitative behavior that is independent of C, as long as $C \gg 1$. All results have a confidence level of 0.99 and a confidence interval of ± 10 percent on the whole distribution.

Fig. 5 shows the histogram of the estimated pdf of the node download times t_i for a network with 10^4 nodes. The distribution architecture is a mesh modeled with CD-CSGP 3. We see that all nodes complete the download in at most one round.

Although distributions like the one depicted in Fig. 5 are the prime output of GROOVER, we show mostly aggregate results (means) in the following, which are more compact and yet convey the fundamental insights of the results that we obtained.

6.1 Tree-Based Distribution Architectures

As we have seen in Fig. 1, the content distribution network is built on top of the overlay network. We start by evaluating the influence of the neighbor set size in the overlay network on the delay in the content distribution network. We see in Fig. 6 that a neighbor set size of $|\mathcal{B}_i| = 8$ is sufficient to achieve as good a performance as for the case where every node in the overlay network knows about every other node, which corresponds to the case $|\mathcal{B}_i| = \mathcal{N}$, $\forall i$.

If the neighbor set is small ($|\mathcal{B}_i| = 4$), the mean download time grows for an increasing number of nodes much faster than for the case where $|\mathcal{B}_i| = \mathcal{N}$.

These observations are valid and are independent of the constraints on the outdegree (results are not shown here for

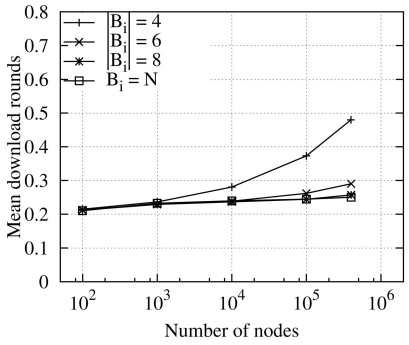

Fig. 6. $\overline{T}$ for different numbers of neighbors $|\mathcal{B}_i|$ in the overlay (CD-CSGP 1 with the outdegree between 1 and 4.)

space reasons) and the type of content distribution tree (hop-driven or cost-driven). For this reason, we assume, for the rest of the paper, that the neighbor set is sufficiently large, i.e., $|\mathcal{B}_i| \geq 8$.

We first focus on the comparison between hop-driven and cost-driven trees and the difference in the way that the trees are built and the different constraints for $k^{\max}$ and $k^{\min}$ have a major impact on the shape of the tree and on the download performance.

Fig. 7 shows the mean download time as a function of the total number of nodes. We see that the mean download time is much lower for cost-driven trees, as compared to hop-driven ones. This is due to the fact that the cost-driven trees use the lowest eligibility time as a criterion for selecting the leaf node to which a new node is attached. Therefore, subtrees with slow nodes being the root will grow much slower, i.e., have very few nodes, as compared to subtrees with fast nodes. On the other hand, hop-driven trees try to keep the step distance $d^{(i)}$ of all leaf nodes approximately the same, which means that subtrees with slow nodes being the root are potentially much larger than in the case of cost-driven trees.

Another parameter that has a major impact on the performance is the outdegree. We first consider the effect of $k^{\max}$: A node i can increase its number of children only if its upload bandwidth is not saturated. This means that if a node has a high upload bandwidth but receives chunks at a low rate, it can serve as many nodes as its maximum outdegree $k^{\max}$ allows. Since, in hop-driven trees, it is more frequent that fast nodes receive at a low rate (this concerns all the fast nodes belonging to a subtree rooted at a slow node), the possibility of increasing the outdegree helps, as it allows fast nodes to use more of their upload bandwidth. In the case of cost-driven trees, fewer fast nodes will receive chunks at a low rate and the benefit of an increased outdegree is not visible.

The impact of the minimum outdegree on the download performance is also very interesting. If we impose a minimum outdegree $k^{\min} = 2$, a slow node will divide its low upload bandwidth available by 2 to serve each of the two children, which will affect the speed of content propagation to the entire subtree rooted at the slow node. If we allow instead a minimum outdegree of 1, in the case of hop-driven trees, this will cut the mean download time in half (see Fig. 7). For cost-driven trees, the value $k^{\min} = 1$ will

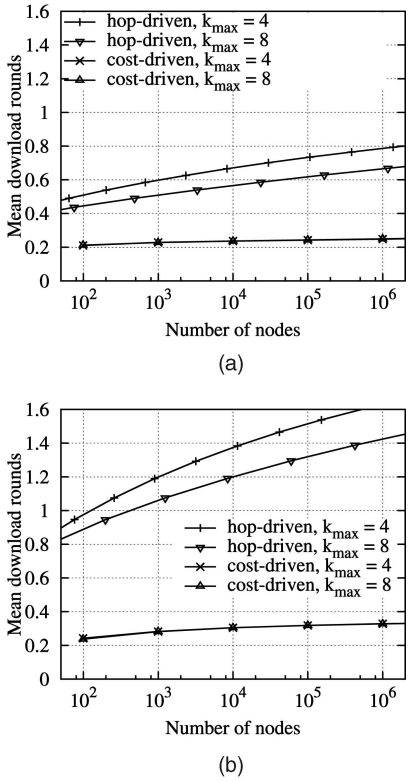

Fig. 7. Mean download time for cost-driven and hop-driven trees. (a) $k^{\min} = 1$. (b) $k^{\min} = 2$.

also reduce the mean download time but not as dramatically as for hop-driven trees. This is expected, since, in cost-driven trees, the overall influence of slow nodes on the download times is reduced, as already explained. To the best of our knowledge, the impact of the minimum outdegree on the download performance has not been discussed previously in the literature.

The superior download performance of cost-driven trees, as compared to hop-driven trees, comes at the cost of a greater step distance. In Fig. 8, we report the distribution of the step distance for different cases. As expected, the mean number of hops in cost-driven trees is higher than in hop-driven trees. However, the individual hops are shorter (smaller chunk download time t^{step}), so the file download time is smaller.

In the case of hop-driven trees, the choice of the minimum outdegree ($k^{\min} = 1$ or $k^{\min} = 2$) has very little influence on the distribution of the step distance, whereas, for cost-driven trees, a value of $k^{\min} = 1$ results in a much larger step distance than a value of $k^{\min} = 2$. In hop-driven trees, the probability of having many consecutive hops with a high rate is very low, so the outdegree of fast nodes increases after a few hops and the total number of nodes reached becomes comparable for $k^{\min} = 1$ or $k^{\min} = 2$.

Finally, we consider the impact of churn on the file download times. We assume that, when a node starts downloading the file, with a probability p_{churn}, it will leave the network during the download. The time until a node leaves is selected uniformly from the interval between the beginning of the download and the estimated end of the

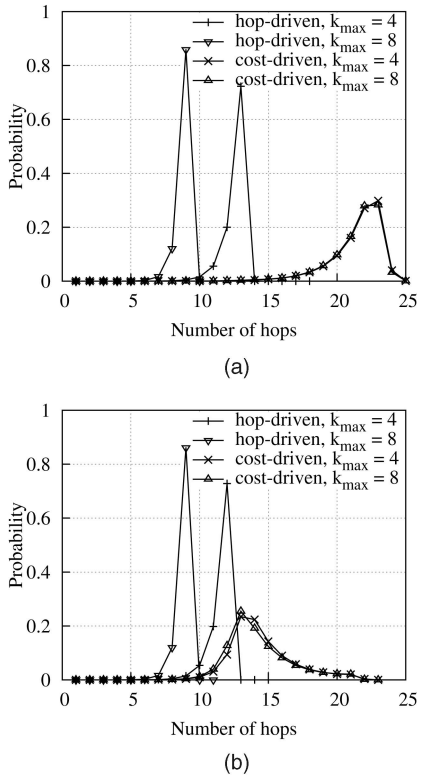

Fig. 8. Distribution of the step distance for 10^6 nodes. (a) Minimum outdegree $k^{\min} = 1$. (b) Minimum outdegree $k^{\min} = 2$.

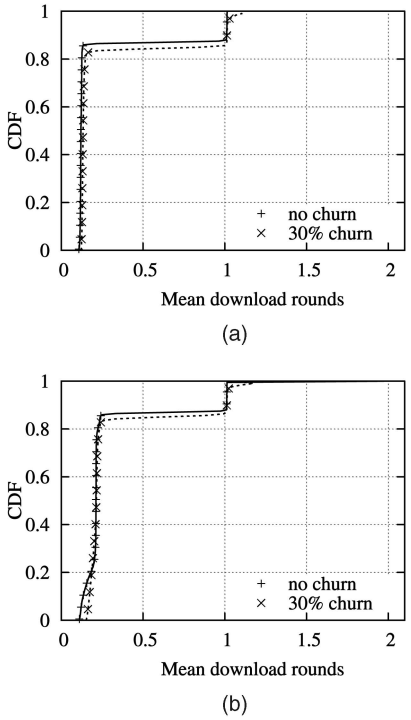

Fig. 9. CDF of the download times for 10^4 nodes with a churn probability of 30 percent. (a) Outdegree: $k^{\min} = 1$ and $k^{\max} = 4$. (b) Outdegree: $k^{\min} = 2$ and $k^{\max} = 4$.

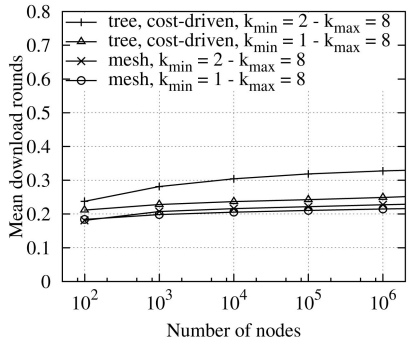

Fig. 10. Mean download time $\overline{T}$ for cost-driven trees and mesh architectures with different outdegree constraints.

download.[6] Fig. 9 shows the cumulative distribution function (CDF) of the download times with different outdegree constraints using a cost-driven tree architecture (we place a subset of the markers to differentiate the curves). The CDF is built by considering the nodes that have completed the download. Even in the presence of 30 percent of nodes leaving, the download performance deteriorates only very little.

We can explain this "surprising" result as follows: In a tree structure, a high percentage of nodes are leaf nodes. However, only internal nodes that leave will impact the performance. The download time depends on the initial delay to get the first chunk and the download rate. When a node i becomes an orphan, since its parent has left, it searches among its neighbors for a node that has upload bandwidth and has downloaded a number of chunks that is greater than the number of chunks owned by the node i. This very simple policy automatically selects the "faster" nodes among the neighbors, preserving the download rate seen by node i. Thus, if the new parent can offer the same download rate, a small increase in the step distance will not have a great impact. Consider also that, according to the distribution of the step distance, most of the nodes have high step distances, so the absolute difference in terms of delay until the first chunk is received is most likely low.

6. We talk here about the estimated download time since the distribution structure is not stable and an ancestor of the node can disappear. This influences the effective download time, so it may happen that a node selects an instant for leaving the network, but in that instant, it has already completed the download (if, for instance, during the download, it has increased its rate).

6.2 Mesh-Based Distribution Architectures

Tree-based architectures have the disadvantage that the leaf nodes do not at all contribute to the distribution of the content. However, when the average outdegree is larger than 2, the leaves make at least 50 percent of all the nodes. It is therefore natural to consider mesh-based architectures. Remember that, to construct a mesh, we first let the diffusion trees reach all the nodes, then nodes with spare bandwidth (typically, the leaves of the diffusion trees) upload to nodes belonging to different diffusion trees. Diffusion trees can be hop-driven or cost-driven. Due to the poor performance of hop-driven trees, we consider meshes with cost-driven diffusion trees only.

In Fig. 10, we compare the mean download time of meshes and cost-driven trees. We see that meshes improve the download time and that the improvement is particularly significant when the minimum outdegree is set to 2. We would like to mention that another advantage of meshes over trees is that meshes are more resilient to node failures.

TABLE 1
Comparison of Cost-Driven Trees and Meshes

Outdegree	#Nodes	Mean Tree Depth	Upload Bandwidth Wasted	
			Cost-driven Tree	Mesh
1 - 8	10^5	17.2	46.9%	13.3%
1 - 8	10^6	21.4	47.5%	13.1%
2 - 8	10^5	12.2	66.2%	26.8%
2 - 8	10^6	14.3	68.9%	29.3%

Table 1 compares meshes and cost-driven trees in terms of the amount of upload bandwidth wasted ($\overline{w^u}$) and also provides information about the mean tree depth, i.e., the average step distance. A mesh architecture reduces, in all the cases that we considered, the wasted upload bandwidth by more than 50 percent. Another interesting result in Table 1 is the increase in the mean tree depth by 40 percent to 50 percent when we reduce the minimum outdegree from 2 to 1. For an operational content distribution system, smaller step distances are attractive: At a given node churn rate, smaller step distances reduce the likelihood that a node will get disconnected due to such an event. If we take into consideration good download performance and robustness in case of node churn, the mesh architecture with a minimum outdegree of 2 is very attractive (see also Fig. 11).

Looking not only at the mean download time but also at the distribution of the download times of individual nodes can give valuable insights. We can, for instance, see up to what degree the slow nodes affect the download times of the faster nodes. We consider that only 13 percent of the nodes are slow nodes. In Fig. 11, we see that, for the cost-driven and mesh architecture, the slow nodes do not negatively affect the download time of the other nodes. On the other hand, in the case of hop-driven trees, the slow nodes negatively affect the download times of about 50 percent of the other nodes.

The performance comparison of the different distribution architectures for a heterogeneous peer population gives some very interesting insights that we have not seen published elsewhere:

- Cost-driven trees and meshes allow us to avoid any negative impact of slow nodes on the download performance of the other nodes. This is due to the fact that the construction process of a cost-driven tree assures that subtrees rooted at a slow node remain very small.
- A minimum node degree of 1 allows a slow node to achieve twice the upload rate to its only child, as compared to the case where a slow node would upload to two children. Although $k^{\min} = 1$ reduces the download times for all the three distribution architectures, the improvement is most pronounced for hop-driven trees.

7 Comparison with Simulations

In order to validate our analytic model and our assumptions, we implement a simple content distribution protocol on top of the PeerSim P2P network simulator [23]. PeerSim

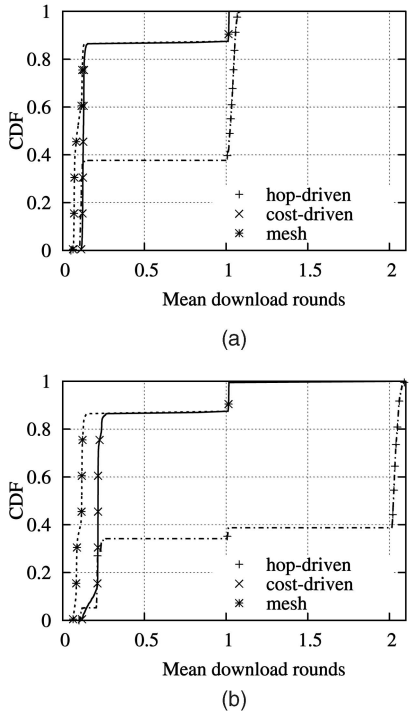

Fig. 11. Comparison of the CDF of the mean download time between a hop-driven tree, a cost-driven tree, and a mesh. (a) Set of 10^5 nodes with outdegree of 1-8. (b) Set of 10^5 nodes with outdegree 2-8.

is a Java-based simulator that consists of many configurable components: It has two types of engines, cycle-based and event-driven, and different modules that manage the overlay building process and the transport characteristics. For a more detailed description of the PeerSim simulator, the interested reader is referred to [23].

7.1 Protocol Description and Simulation Setup

We implemented two file distribution protocols by using the event-driven engine: One represents CD-CGSP 1 (cost driven) and the other represents CD-CGSP 3 (mesh). In the following, we give a high-level view of the protocol messages and procedures, leaving out details about the management of all the situations. The implementation is available at [24]:

- Messages
  - *New Content*. With this message, a node informs its neighbors that it has finished downloading a chunk of a new file and can upload it to interested nodes.
  - *Reply*. With this message, a node replies to a New Content message saying that it is interested, specifying its bandwidth.
  - *Start Downloading*. With this message, a node informs its children that they can start downloading the content previously announced and the rate that they can use.
- Procedures
  - *End First Chunk*. When a node terminates to download the first chunk, it sends out the New Content messages to its neighbors.

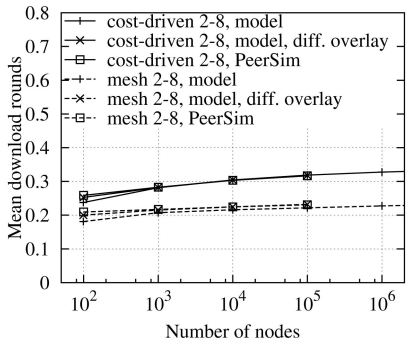

Fig. 12. Comparison between results obtained with GROOVER and with the PeerSim simulator (outdegree of 2-8).

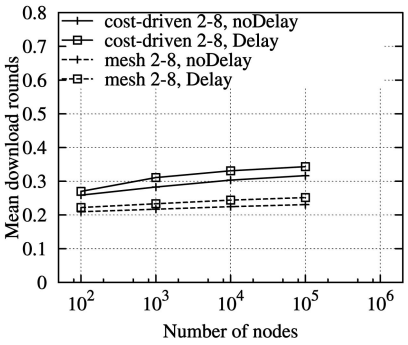

Fig. 13. PeerSim results with a delay uniformly distributed between 0 and 1/10 chunk time (slower class, outdegree of 2-8).

– *Process Replies*. While receiving the replies after a New Content message from its neighbors, a node selects the neighbors that replied as its children until its bandwidth is saturated. Additional replies are ignored.

For the overlay graph construction in PeerSim, we build a random graph with a mean degree of 20. At the transport layer, each message experiences an end-to-end delay that is uniformly distributed between minimal and maximal values. The other constraints concerning the minimum and maximum number of children, number of chunks to be transmitted, and number of diffusion subtrees for the mesh case are the same as before.

7.2 Simulation Results

For a given set of configurations and cost-driven and mesh architectures, we compare the results obtained by using GROOVER and using PeerSim. Due to the computational limits of PeerSim, we were only able to analyze systems with up to 10^5 nodes, even with a more powerful machine than the one used for running GROOVER. For all the results, we computed the confidence intervals for a confidence level of 95 percent. We obtained interval widths of less than 1 percent of the point estimate, not shown in the figures. We first start analyzing the case with no message delay. Fig. 12 shows the case with outdegree of 2-8 (we obtain similar results, which are not reported here for space constraints, with outdegree of 1-8) for cost-driven and mesh architectures. As you can see, the results confirm our analysis. The slight differences between some of the results, especially for a small number of nodes, are due to the properties of the overlay network connectivity: In GROOVER, we assume a hypercubic connectivity with a fixed degree, whereas the simulator constructs a random graph. We have also implemented the random graph in GROOVER and find a better agreement. Overall, we see that analytic and simulation results agree very closely.

To check modeling assumptions, we add a delay in the message transfer in the simulations that is uniformly distributed between 0 and 1/10 of the time necessary to upload a chunk with the minimum bandwidth.[7] Fig. 13 shows the results for different architectures. The delay slightly increases the mean download time, but the quantitative behavior of the system remains the same.

From the comparison between the simulation and analytic results, we can conclude that our analytic model is able to capture the essential performance characteristics of the file distribution architectures, provided that the mean message delay is not exceedingly large with respect to chunk transmission. With the chunk size in the order of a few tens of kilobytes, this assumption is realistic.

8 Practical Aspects and Further Extensions

We have studied three different content distribution architectures and evaluated their performance. We think that our results can help in improving the design of content distribution architectures. We therefore discuss briefly how the systems that we describe with our model can be put into practice in a real content distribution system, where the distribution graph is constructed by a *distributed* algorithm and where some of the assumptions made (for example, that the upload/download bandwidth of the node is known) may not hold. In the following, we will sketch out the basic operation of such a distributed algorithm for mesh construction, as defined in CD-CSGP-3, which is, in fact, the most promising architecture, since it best exploits the resources and offers a resilient structure in case of bandwidth fluctuations.

Our results show that the minimum outdegree represents an important configuration parameter: Setting k^{min} to 1 helps in improving the performance independently of the maximum outdegree. We assume that the overlay layer (see Fig. 1) has been constructed, and each node i knows its neighbors $\mathcal{B}_i$. Assume that the root has been selected and the root has chosen some of the nodes in its neighbor set to which it starts transmitting chunks.

If a node i gets contacted and starts receiving the first chunk, it will wait until this chunk is completely received. At that point, node i will select among its neighbors a subset of candidates. Node i will contact the nodes in $\mathcal{B}_i$ and retain the ones that have not yet received chunks from any other node in its candidate set $\mathcal{C}_i \subseteq \mathcal{B}_i$. Let us assume for the moment that node i knows its upload bandwidth b_i^u and the download bandwidths b_j^d, $j \in \mathcal{C}_i$. Node i will open connections to its neighbors in $\mathcal{C}_i$ until it is saturated (that is, its upload bandwidth is all used up or the constraint on the maximum number of outgoing connections is reached). If

7. This corresponds to two times the time necessary to upload the chunk with the highest bandwidth, so it is a large delay.

the node is not able to find untouched nodes in $\mathcal{B}_i$, it will search for neighbors that belong to different diffusion trees.

There are quite a few cases where the upload and download bandwidth may, in fact, be known; for instance, in a corporate environment or even in an Internetwide P2P system. Today, many file-sharing tools such as BitTorrent or Edonkey allow the user to *limit* the upload bandwidth. In this case, one can set both the available upload and download bandwidth to that value.

However, if the available upload and download bandwidths of node i are not known, node i could proceed as follows: It first opens only k^{min} connections and starts transmitting to k^{min} neighbors. If, after some time, node i observes that its upload bandwidth is not fully utilized, it can send a message searching for a candidate node j whose download bandwidth is not fully utilized down the subtree rooted at node i. The tree would be reorganized as follows: Node j disconnects from its current parent and becomes a direct child of node i.

8.1 Possible Extensions

We have presented a technique that allows us to evaluate different content distribution architectures. In this paper, we have limited ourselves for space reasons to the case where the upload and download bandwidth between neighbor nodes do not change during the whole transfer. However, the technique that we have introduced can be extended to handle more complex scenarios and we have already made extensions for bandwidth fluctuations during the distribution of the file. We have also been able to extend our technique to evaluate *real-time* video streaming architectures. For details, see [21].

9 Conclusions

The use of trees for content distribution has been studied intensively in the literature, but fundamental issues such as bandwidth heterogeneity and varying node outdegree, as well as different minimum and maximum outdegrees, have received very little attention. In this paper, we have made several contributions concerning 1) the formalism to describe the content distribution graph, 2) the methodology to compute the relevant performance metrics, and 3) the performance results.

We have defined a new analytical methodology called CSGPs that is suitable for the description of content distribution architectures based on trees or meshes. The formalism specifies the evolution of the content distribution process as a semi-Markov process. To the best of our knowledge, SGPs were only used to study connectivity properties and have not been applied to the performance analysis.

We have developed a numerical solver GROOVER that emulates a random walk on the DTMC embedded in the semi-Markov process. GROOVER allows for a very efficient computation of the performance metrics, even for a very large number of nodes, which is not feasible by using a standard discrete-event simulation. On a PC, we can use GROOVER to evaluate distribution architectures of a million of nodes and more in just a few hours, which makes our approach very attractive for the evaluation of large P2P files distribution systems.

Using GROOVER, we have obtained a number of novel results:

1. Cooperative content distribution networks are inherently *self-scalable* in that they take advantage of the resources provided by every peer. Millions of nodes can be reached by a content within a mean download time that is near the download time in a one-to-one communication.
2. Cost-driven trees are able to obtain performances that are comparable to mesh-based architectures since they succeed at placing slow nodes mainly at the leaves. Besides resiliency, the gain of meshes is in reducing the download time of high bandwidth nodes.
3. The minimum node outdegree represents an important design parameter that has not been considered before. Allowing for a minimum node outdegree of 1, as compared to 2, helps in decreasing the download time. Thus, applications such as eMule that force a minimum outdegree of 2 are not able to fully exploit the resources.

Acknowledgments

The authors would like to thank P. Brown, A. Chaintreau, T. En-Najjary, and A. Legout for their helpful comments on earlier versions of this paper, as well as the anonymous reviewers for their constructive comments. This work was supported in part by the European Union under the E-NEXT Project FP6-506869. The final part of the work in Trento was also supported by the Progetti di Ricerca di Interesse Nazionale (PRIN) project PROFILES.

References

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

[2] X. Yang and G. de Veciana, "Service Capacity of Peer-to-Peer Networks," *Proc. IEEE INFOCOM,* 2004.

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

[4] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," *Proc. ACM SIGCOMM '04,* Sept. 2004.

[5] B. Cohen, "Incentives Build Robustness in BitTorrent," *Proc. First Workshop Economics of Peer-to-Peer Systems (P2P-Econ '03),* June 2003.

[6] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," *Proc. Third Usenix Symp. Internet Technologies and Systems (USITS '01),* Mar. 2001.

[7] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: Highbandwidth Multicast in a Cooperative Environment," *Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03),* Oct. 2003.

[8] E.W. Biersack, D. Carra, R. Lo Cigno, P. Rodriguez, and P. Felber, "Overlay Architectures for File Distribution: Fundamental Performance Analysis for Homogeneous and Heterogeneous Cases," *Computer Networks J.,* vol. 51, no. 3, pp. 901-917, Feb. 2007.

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

[10] S. Nikoletseas, J. Reif, P. Spirakis, and M. Young, "Stochastic Graphs Have Short Memory: Fully Dynamic Connectivity in Poly-Log Expected Time," *Proc. 22nd Int'l Colloquium Automata, Languages, and Programming (ICALP '95),* pp. 159-170, 1995.

[11] P. Erdös and A. Renyi, "On Random Graphs," *Publicationes Mathematicae,* vol. 6, pp. 290-297, 1959.
[12] D. Carra, R. Lo Cigno, and E.W. Biersack, "Content Delivery in Overlay Networks: A Stochastic Graph Processes Perspective," *Proc. 49th Ann. IEEE Global Telecomm. Conf. (GLOBECOM '06),* 2006.
[13] D. Carra, "Performance Evaluation of Overlay Content Distribution Systems," PhD dissertation, Univ. of Trento, Mar. 2007.
[14] D. Carra, R. Lo Cigno, and E.W. Biersack, "Fast Stochastic Exploration of P2P File Distribution Architectures," *Proc. 49th Ann. IEEE Global Telecomm. Conf. (GLOBECOM '06),* 2006.
[15] S. Keshav, *An Engineering Approach to Computer Networking.* Addison-Wesley, 1997.
[16] C. Gkantsidis and P. Rodriguez, "Network Coding for Large-Scale Content Distribution," *Proc. IEEE INFOCOM,* Mar. 2005.
[17] J. Honerkamp, *Stochastic Dynamical Systems: Concepts, Numerical Methods, Data Analysis.* Wiley-VCH, 1994.
[18] D.T. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *J. Physical Chemistry,* vol. 63, no. 25, pp. 2340-2361, 1977.
[19] S.M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists,* eighth ed. Academic Press, Dec. 2002.
[20] R. Gaeta, M. Gribaudo, D. Manini, and M. Sereno, "Analysis of Resource Transfer in Peer-to-Peer File Sharing Applications Using Fluid Models," *Performance Evaluation: An Int'l J. Peer-to-Peer Computing Systems,* vol. 63, no. 3, pp. 147-264,
[21] D. Carra, R. Lo Cigno, and E.W. Biersack, "Graph Based Modeling of P2P Streaming Systems," *Proc. Sixth IFIP Int'l Conf. Networking (Networking '07),* May 2007.
[22] *GROOVER,* http://dit.unitn.it/networking/tool/groover/, Feb. 2007.
[23] *PeerSim: A Peer-to-Peer Simulator,* http://peersim.sourceforge.net/, Feb. 2007.
[24] *File Distribution Protocol in PeerSim,* http://dit.unitn.it/networking/tool/cdp-peersim/, Feb. 2007.

Damiano Carra received the Laurea degree in telecommunications engineering from Politecnico di Milano in 2000 and the PhD degree in computer science from the University of Trento in 2007. From 2000 to 2003, he worked as a technical consultant for leading industries in telecommunications and computer science. His research interests include the modeling and performance evaluation of peer-to-peer networks.

Renato Lo Cigno received the Dr. Ing. degree in electronic engineering from Politecnico di Torino in 1988. He is an associate professor in the Department of Computer Science and Telecommunications (DIT), University of Trento, Italy, where he is one of the founding members of the Networking Research Group. From 1999 to 2002, he was with the Telecommunication Research Group, Electronics Department, Politecnico di Torino. From June 1998 to February 1999, he was with the Computer Science Department, University of California, Los Angeles (UCLA), as a visiting scholar under Grant CNR 203.15.8. He is an area editor of *Computer Networks* (Elsevier) and has served as the chair or a technical program committee member of several leading conferences, including IEEE INFOCOM, the IEEE Global Telecommunications Conference (GLOBECOM), the IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), the IEEE International Conference on Mobile Ad Hoc and Sensory Systems (MASS), and the ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH). His current research interests include the performance evaluation of wired and wireless networks, including mesh networks and overlay, peer-to-peer systems modeling, simulation techniques and modeling, and resource management and congestion control. He is a coauthor of more than 100 journal papers and conference proceedings in communication networks and systems. He is a member of the IEEE and the ACM.

Ernst W. Biersack received the MS and PhD degrees in computer science from the Technische Universität München, Munich, and the habilitation degree from the University of Nice, France. Since March 1992, he has been a professor of telecommunications at Institut Eurecom, Sophia Antipolis, France. His current research interests are peer-to-peer systems and network tomography.