Graph Based Analysis of Mesh Overlay Streaming Systems

Damiano Carra, Renato Lo Cigno, and Ernst W. Biersack

***Abstract*—This paper studies fundamental properties of stream-based content distribution services. We assume the presence of an overlay network (such as those built by P2P systems) with limited degree of connectivity, and we develop a mathematical model that captures the essential features of overlay-based streaming protocols and systems.**

The methodology is based on stochastic graph theory, and models the streaming system as a stochastic process, whose characteristics are related to the streaming protocol. The model captures the elementary properties of the streaming system such as the number of active connections, the different play-out delay of nodes, and the probability of not receiving the stream due to nodes failures/misbehavior. Besides the static properties, the model is able to capture the transient behavior of the distribution graphs, i.e., the evolution of the structure over time, for instance in the initial phase of the distribution process.

Contributions of this paper include a detailed definition of the methodology, its comparison with other analytical approaches and with simulative results, and a discussion of the additional insights enabled by this methodology. Results show that mesh based architectures are able to provide bounds on the receiving delay and maintain rate fluctuations due to system dynamics very low. Additionally, given the tight relationship between the stochastic process and the properties of the distribution protocol, this methodology gives basic guidelines for the design of such protocols and systems.

***Index Terms*—Modeling, Stochastic processes, Performance analysis, Simulation, Peer-to-Peer, Streaming, Graph-based protocols**

I. Introduction

The recent success of streaming based on peer-to-peer (P2P) applications seems to achieve what traditional streaming and multicasting applications have never achieved: distributed video-on-demand and live broadcasting on the Internet. Tree based systems [1][2][3] that have been proposed earlier coexist now with more advanced mesh-based systems [4][5][6] that are more resilient to node dynamics and bandwidth variations as seen in the Internet.

In spite of the success of P2P streaming, the fundamental properties of such systems have not been investigated in depth (see Sect. I-A for a discussion of existing works). In particular, we are not aware of any study concerning the behavior of the streaming distribution system as a function of the topological properties of the graph that is built by the P2P application for the

D. Carra and R. Lo Cigno are with Dip. di Informatica e Telecomunicazioni, Università di Trento, Trento, Italy, E-mail [carra, locigno]@dit.unitn.it.

E. W. Biersack is with Institut EURECOM, Sophia Antipolis, France, E-mail: erbi@eurecom.fr.

D. Carra is currently working at Institut EURECOM, Sophia Antipolis with a Post-Doc position partly financed by EU Project CASCADAS (IST-027807).

This work was partially supported in Italy by the Italian MIUR PRIN project PROFILES and in France by the project CASCADAS (IST-027807) funded by the FET Program of the Eurepean Commission.

information transfer.

In this work we develop a mathematical model based on stochastic graph theory that can be used to analyze fundamental performance issues of overlay streaming services. The level of abstraction of the model allows to study the fundamental behavior under different conditions, yet maintaining a limited complexity. Many studies analyze a *static* graphs that captures the properties of a snapshot of the network (see [7] and the references therein). In our work, instead, we study the *dynamics* of the graphs, i.e., the evolution of the structure over time.

We derive the master equations (MEs) that define the evolution of the streaming system in time. The MEs take into account the fundamental characteristics of the streaming protocol as well as the bandwidth available at nodes for the streaming application. The model allows assessing the impact of different protocol choices and of bandwidth heterogeneity on the delivery process and provides insights in how to improve existing streaming strategies.

A fast and effective Monte Carlo integration methodology is used to solve the mathematical model. The solution provided by this method is compared with other modeling techniques to show the flexibility and computational efficiency of that approach.

The results obtained by the systematic study of different configurations show that performance is mainly influenced by the policies related to content format (how much redundant information is sent). Mesh based architectures are very robust to failures, even in presence of high churn and the delay experienced by nodes stays bounded.

A. Related Work

In the last few years many solutions have been proposed for overlay streaming services. Such systems can be classified according to the basic structure they use to deliver the content.

Systems such as ALMI [1], NICE [2] and Zigzag [3] organize the nodes in a tree structure. The stream is received from a single parent and uploaded to a set of children (except for the leaves). The differences among these systems concerns the algorithms used for adapting the tree in face of the node dynamics. Systems such as Narada [4], Coolstreaming [5] and PULSE [6] build a mesh structure. Nodes download from a set of nodes, called parents, which can change over time. Problems related to delay and synchronization are handled according to different heuristics. Other systems adopt an hybrid approach (e.g., SplitStream [8], PRIME [9], Multi-tree ESM [10]), where the stream is distributed using multiple trees obtaining a structured mesh.

The above distribution protocols were not designed with a performance driven approach as far as delivery is concerned. Many proposals use heuristic methods to improve performance, but these heuristics are validated a posteriori and protocol parameters are tuned according to these results. Performance analysis of overlay streaming systems received some attention only recently. Most of the analytical works focus on tree based structures (e.g., [11], [12], [13]) or on a specific system (e.g., [14]), but, to the best of our knowledge, no study has been done on modeling general mesh-based streaming systems. Only [15], [16] and [17] start analyzing such systems, but [15] and [16] consider fluid analysis, finding the conditions under which the system can work and [17] adopts a simulative approach.

Some recent studies on graph theory analyze the properties of growing networks [7][18], but the way the graph structure can grow is not constrained by protocol rules, thus resulting of little use in the analysis of highly structured systems. Our model considers also the

properties of the overlay graph as well as the access bandwidth variability, and the protocol rules that drive the graph evolution. In this sense our approach extends the general concepts of these theoretical studies to the specific problem considered.

In the preliminary version of this paper presented in [19] we carried out an initial analysis of overlay streaming systems. Compared to that paper, this work presents the theory and methodology of the analytical model and it explains in details how the Markov process describes the evolution of the system over time. Moreover, we implement an overlay streaming protocol using the PeerSim P2P simulator [27], validating the analytical results with the simulative ones.

II. Mesh-based Overlay Streaming Systems

We do not consider here any specific system, but we identify *common basic characteristics* of recent proposals, focusing on hybrid approaches, such as [5][9][10], where the mesh is built as superposition of trees, obtaining a *structured mesh*. Consider an overlay network built by a P2P application. Once the overlay layer is built, paths between the source and the destinations are created following the rules of the distribution protocol. At each hop, nodes both receive the stream and contribute uploading it to other nodes, i.e., they work as content relay. Since nodes in such networks can appear or disappear frequently, the set of nodes from which a node is downloading changes over time.

A. System parameters

The content is distributed using R different stripes. Each stripe contains part of the stream (coded, for instance, using MDC techniques [20]). A node needs $R' < R$ out of R stripes to achieve a target quality, while the remaining $R - R'$ stripes contain redundant information. We assume that each node downloads a specific stripe R_i from a single node. Downloading the same stripe from multiple parents does not increase the quality of the received stream. Moreover, each node downloads only a single stripe from a given parent, even if the parent could provide multiple stripes, which limits the impact of a parent that leaves.

Even if the structure is a mesh, looking at the system at a specific instant t, it is possible to identify sub-structures inside the mesh. If we consider the graph at time t and we consider the nodes that are downloading stripe R_i, it is possible to construct a tree that connects these nodes. We call such a tree "diffusion tree." The whole mesh can be seen as a set of overlapping diffusion trees [17], which change over time.

The evolution of the network is subject to two main events: node arrivals and departures. We assume that arrivals and departures are exponentially distributed according to rates $\lambda(t)$ and $\mu(t)$ respectively. The dependence on time makes the model more flexible: for instance, different arrival patterns, such as flash crowds or more smooth arrivals, can be described. Let T_{str} be the duration of the stream and N the mean number of nodes receiving the stream at steady state. We consider a situation where a fraction of the nodes joins the stream at time zero, and there is an *arrival interval* during which $\lambda(t) > \mu(t)$ until steady state is reached. Fig. 1 shows a sample arrival pattern.

The departure rate $\mu(t)$ is the inverse of the mean time spent in the system (sojourn time). $\lambda(t)$ at steady state compensates departures. For a given time interval T, the ratio between the cumulative number of nodes that left and the mean number of active nodes during T is defined as the *churn* of the system. A 100% churn means that during T the number of the departed nodes is equal to the mean number of nodes in the system, i.e., there is

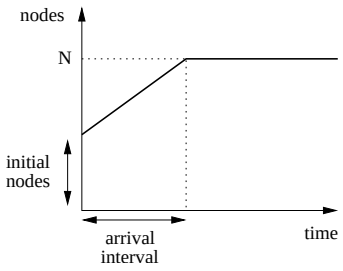


Fig. 1. Sample arrival pattern of nodes joining a stream.

on average a complete change of the nodes during T.

Nodes are divided into different classes according to their bandwidth. Each class j has an upload bandwidth $b_u^{(j)}$ and a download bandwidth $b_d^{(j)}$, which can be either symmetric, asymmetric or correlated, e.g., $b_i^u + b_i^d$ constant, as in a shared medium access. The bandwidths are random variables described by a probability density function (pdf) that is known (e.g., derived from measurement studies).

The rate of the streaming is r_{str}. We suppose that all nodes have a download bandwidth at least equal to the streaming rate. Each stripe has a rate equal to r_{str}/R', and we assume that the server is able to upload all the R stripes, i.e., it has a bandwidth greater than Rr_{str}/R'. Each node has a constraint on maximum and minimum number of active uploads that limit the possible outdegree of the node: $k^{\max}$ is the *maximum outdegree* and and $k^{\min}$ is the *minimum outdegree*.

Each node has B overlay neighbors. Among its neighbors the node selects its *parent* nodes, i.e., those from which it downloads. R' parents are called *active*; the remaining are called *standby*, since they are used as a backup in case of an active parent failure[1].

[1]We borrowed the concept of grouping active and standby parents from [21].

B. Join, Update and Leave Procedures

Nodes belonging to the initial set start building a diffusion tree for each stripe. The number of nodes in each diffusion tree depends on the characteristics of the nodes involved such as the bandwidth. Each node is involved in multiple diffusion trees.

When a new node arrives, it randomly chooses an active node as first contact and then builds its neighbor list with the help of this node. From the neighbor list, the node selects its parents and connects to them.

With rate λ_{up} nodes periodically search among their neighbors for new connections in order to increase their indegree. For standby parents, the bandwidth is not reserved, so the total number of parents can exceed the ratio between the stripe rate and the node download bandwidth.

When a node leaves, all the inbound and outbound connections are canceled. Orphan nodes try to replace the parent that has left. If the parent that has left was in the standby set, the node does not react (it simply loses a backup parent). If the parent that has left was in the active set, the node tries to switch the state of a standby parent, i.e., it starts downloading from the standby parent that has enough available upload bandwidth. If a node has no backup parents, there will be a temporary loss of quality whose extent depends on the time necessary to search for a new parent. The node can either try to look for a new active parent immediately or wait for the next *Update* procedure. In order to keep the model simple, we consider only the latter case. In Sect. VI we analyze with simulations the difference between these two cases.

III. System Model

The network of contacts among users of a P2P networks can be modeled as a graph, where nodes represent the users and edges the neighborhood relationship. When

the users start exchanging data (in our case, they start receiving and distributing the stream) they use a subset of the available outgoing/incoming edges. The number of neighbors that are uploading to a node, for instance, represents the number of parents from which the node downloads the content, and it can be considered as a metric to measure the total rate received, and consequently the quality of the streaming. The focus of our analysis is the characteristics of the *distribution graph* (see Fig. 2), i.e., the subgraph of the overlay graph, where edges are the connections effectively used by nodes.

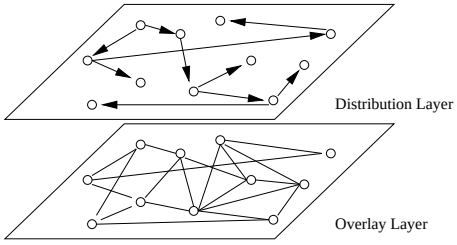


Fig. 2. Overlay and distribution graphs

In general, the distribution graph is time varying, i.e., nodes and edges can appear or disappear in time. The evolution of the graph can be seen as a stochastic process with Markovian properties, since the graph at time $t+dt$ depends only on the graph at time t and the event (join or leave of a node) occurred during dt.

A. Formal Description of the System

Considering the system at a specific instant t, we can identify for each node two types of relationship: parents (or, equivalently, children) and neighbors. For parents we can associate an identifier of the stripe exchanged — different identifiers are selected in case of active or standby stripe.

Let α_k and σ_k be the identifier of stripe k, when it is active and standby respectively. Let η be the identifier that indicates that two nodes are neighbors in the overlay. The overlay graph and the distribution graph can thus be described by the connectivity matrix $\boldsymbol{S}$, where each element s_{ij} describes the relationship between node i and node j:

$$s_{ij} = \begin{cases} 0 & \text{nodes } i \text{ and } j \text{ are not neighbors} \\ \eta & \text{nodes } i \text{ and } j \text{ are neighbors and do not exchange stripes} \\ \alpha_k & \text{nodes } i \text{ and } j \text{ are neighbors and } i \text{ is } \textit{active} \text{ parent of } j \text{ with stripe } k \\ \sigma_k & \text{nodes } i \text{ and } j \text{ are neighbors and } i \text{ is } \textit{standby} \text{ parent of } j \text{ with stripe } k \end{cases} \quad (1)$$

Since each parent can upload only a *single stripe* to a node, s_{ij} can assume only the above values. Along with the connectivity matrix, each node has a upload bandwidth that can be represented, for a given stripe rate, as the maximum number of active children a node can have.

State transitions are determined by the events join, leave, and update described in Sect. II. We assume that the rates of these events are exponentially distributed with parameters λ, μ and λ_{up} respectively. For each event it is possible to find the transition probabilities from a state $\boldsymbol{S}$ to a state $\boldsymbol{S'}$ that describes the new connectivity matrix. We will see that each event corresponds to a set of operations. As a consequence, the corresponding transition is not simple and more than one element in the connectivity matrix may change.

Join and Update

The join procedure is composed by two steps: node arrival and connection to stripes, with the latest being equivalent to an update procedure.

In the arrival procedure the arriving node b builds the neighbor set. The number of initial neighbors is equal to

B. Since neighbors are randomly chosen, the new state will have a new row and a new column filled with η for each neighbor relationship. The possible transitions are given by the combination of B elements out of N total nodes with equal probability.

In the update step, a node receives from each neighbor i a vector $V_{i,b}$ containing the (active) stripes the neighbor can provide. The vector has R elements and element k contains the value of α_k if the neighbor has the stripe, otherwise it contains zero. Within each of the vectors received, the node selects randomly one of the available stripes, independently (and asynchronously) for each parent. Given the vectors, it is possible to build all the possible combinations of stripes downloadable from the whole neighbors' set (the possible combinations include the different order of arrival and selection of parents). Since the selection is random and done independently neighbor by neighbor, each combination has equal probability. In the Appendix A we give the procedure that finds all the possible combinations. In the following we present a small example with $R = R' = 3$, $\alpha_k = \alpha_1, \alpha_2, \alpha_3$ and a network with 5 nodes. Assume that node 5 has just arrived and built its neighbor set: the overlay and the diffusion trees are the ones depicted in Fig. 3.

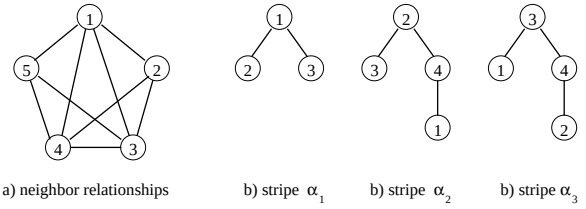


Fig. 3. Neighbor relationships and diffusion trees of the example when node 5 joins the stream.

The connectivity matrix built upon node 5 joining the stream is represented in Fig. 4, where column 5 identifies that node 5 is still not receiving any stripe.

$$\boldsymbol{S} = \begin{bmatrix} 0 & \alpha_1 & \alpha_1 & \eta & \eta \\ \eta & 0 & \alpha_2 & \alpha_2 & 0 \\ \alpha_3 & \eta & 0 & \alpha_3 & \eta \\ \alpha_2 & \alpha_3 & \eta & 0 & \eta \\ \eta & 0 & \eta & \eta & 0 \end{bmatrix}$$

Fig. 4. State reached after node 5 has joined the overlay.

From the definitions in (1), row b represents the children of node b, while column b represents the parents of node b. Node 5 now receives from its neighbors the following vectors: $V_{1,5} = [0, \alpha_2, \alpha_3]$ from node 1, $V_{3,5} = [\alpha_1, \alpha_2, 0]$ from node 3, and $V_{4,5} = [0, \alpha_2, \alpha_3]$ from node 4. Table I reports in the first row the three vectors $V_{i,5}$ received by node 5 from its neighbors 1, 3, and 4. The following rows in Table I are the four possible combinations (see Appendix A for details) of downloadable stripes.

The selection strategy depends on the protocol. Here for the sake of simplicity we assume that the node b selects randomly the stripe within the vector $V_{i,b}$ immediately upon receiving it, so the combinations are all equal likely.

TABLE I
STRIPES NODE 5 CAN SELECT

$[0, \alpha_2, \alpha_3]$ from node 1	$[\alpha_1, \alpha_2, 0]$ from node 3	$[0, \alpha_2, \alpha_3]$ from node 4
α_2	α_1	α_3
α_3	α_1	α_2
α_3	α_2	0
0	α_2	α_3

Supposing that node 5 selects the last row of Table I, $[0, \alpha_2, \alpha_3]$ (i.e., node 5 will download the stripe α_2 from

node 3 and the stripe α_3 from node 4) then the following state will be $\boldsymbol{S'}$ represented in Fig. 5.

$$\boldsymbol{S'} = \begin{bmatrix} 0 & \alpha_1 & \alpha_1 & \eta & \eta \\ \eta & 0 & \alpha_2 & \alpha_2 & 0 \\ \alpha_3 & \eta & 0 & \alpha_3 & \alpha_2 \\ \alpha_2 & \alpha_3 & \eta & 0 & \alpha_3 \\ \eta & 0 & \eta & \eta & 0 \end{bmatrix}$$

Fig. 5. State reached after node 5 has selected the stripes.

The transition probability is $P(\boldsymbol{S}, \boldsymbol{S'}) = \frac{1}{4}$. Defining the other 3 possible states is trivial. The last row of $\boldsymbol{S'}$, without α_k or σ_k identifies 5 as a leaf node, a state (of the node) that can be left only as a consequence of update procedures of other nodes.

Leave

The leave procedure involves multiple interactions of several nodes. When node b leaves the stream it disappears from all neighbors: the connectivity matrix $\boldsymbol{S'}$ will have all zeros in row and column b.

Each node i that was receiving an active stripe k from the node that has left tries to switch to a standby parent j (that has stripe h): the corresponding value of the connectivity matrix will change from σ_h to α_h. This is done if the standby node j can upload more stripes. Otherwise the corresponding value is switched from σ_h to zero. The identifiers α_k or σ_k of the disappeared stripe will be set to zero wherever it appears in row i. The grandchildren of the node that has left will then try to switch standby parents too. At the end of this composition of steps, the connectivity matrix reaches a stable state that is the final transition. Note that each step is independent, so the final transition probability is easily computed.

B. Applicability of the Model

The system described in Sect. II is not meant as a proposal for a new P2P streaming protocol, but it is an abstract representation of the key features of many protocols like [5][9][10]. In fact all these system share a common idea of "striping" the distribution and building a structured mesh composed of the distribution trees of the single stripes. Moreover, the choice of the neighborhood and of the parents is, up to a certain degree, random, so that considering a purely random choice captures the common behavior and represents (most probably) a lower bound of performances.

The description of the overlay graph through the connectivity matrix is able to capture the essential features of overlay streaming protocols. The details of a protocol may influence different aspects of the model: (i) a protocol may impose different constraints on the structure (e.g., maximum number of neighbors, i.e. maximum number of non null elements per row); (ii) it may influence, given a state, the possible states that can be reached; (iii) it may change the transition rates. Nevertheless, the basic structure of the proposed model, the connectivity matrix and its evolution according to the protocol policies, remains unchanged. By properly translating the protocol policies and constraints into the connectivity matrix properties and transition rates, the methodology is able to provide insights into the fundamental performances that can be obtained by different overlay protocols.

C. Master Equations

The evolution of the graph that describes the overlay streaming systems is a Markov process with state space and transitions defined above. The temporal behavior can be described using the differential form of the Chapman-Kolmogorov equations, known as *Master Equations*

(MEs) [7].

Let $P(\boldsymbol{S_i}, t)$ be the probability to be in state $\boldsymbol{S_i}$ at time t. The variation of the probability $P(\boldsymbol{S_i}, t)$ in time can be expressed as

$$\frac{\partial}{\partial t} P(\boldsymbol{S_i}, t) = \sum_{\boldsymbol{S_j}} w_{\boldsymbol{S_j}, \boldsymbol{S_i}}(t) P(\boldsymbol{S_j}, t) \qquad (2)$$

where $w_{\boldsymbol{S_j}, \boldsymbol{S_i}}(t)$ represents the transition rates from the state $\boldsymbol{S_j}$ to the state $\boldsymbol{S_i}$ at time t. The general formulation of the Master Equations must be specialized for our problem: the transition rates are closely related to the streaming protocol policies and can be found as described in the previous section.

D. Distribution Graph Properties

The information given by the MEs are relative to the whole overlay and distribution graph. In order to analyze the system independently from the size of the network, it is useful to reorganize the information contained in state $\boldsymbol{S}$. We consider the distribution of two main performance indexes that summarize the structural characteristics of a distribution graph: the degree distribution and the delay distribution [7].

The degree distribution $P_b(d, t)$ is the probability that node b has d connections at the distribution layer at time t. We can identify both indegree and outdegree distributions ($P_b(d_i, t)$ and $P_b(d_o, t)$ respectively, with $d_i + d_o = d$), that represent the number of children and the number of parents of node b. The outdegree (indegree) distribution of a node b is derived from the correspondent row (column) b of the connectivity matrix $\boldsymbol{S}$. We can also determine the total degree distribution defined as

$$P(d, t) = \frac{1}{N(t)} \sum_{b=1}^{N(t)} P_b(d, t) \qquad (3)$$

where $N(t)$ is the number of nodes attached to the stream at time t.

The delay distribution represents the distance of the node from the source of the stream considering all the active stripes the node is receiving. We define $P_b(\ell, t)$ as the probability that node b is ℓ hops away from the source at time t, where ℓ is the maximum among all the stripes. Similarly to the degree distribution, we can derive the total delay distribution

$$P(\ell, t) = \frac{1}{N(t)} \sum_{b=1}^{N(t)} P_b(\ell, t) \ . \qquad (4)$$

The probability $P_b(\ell, t)$ can be obtained from $\boldsymbol{S}$ in the following way: given a node b and a stripe k, we can recursively find the parent that is providing the stripe (through σ_k or α_k), counting the number of recursions, until we reach the root. In Appendix A we give the procedure that can be used to obtain the number of steps from $\boldsymbol{S}$.

E. Rate Equations

The MEs fully determine the evolution in time of the stochastic system. Considering the degree and delay distributions, it is also useful to have the equations for the average value. The correspondent equations are called *Rate Equations* (REs):

$$\frac{\partial}{\partial t} \overline{d} = \frac{\partial}{\partial t} \sum_{d} d P(d, t) \qquad (5)$$

The REs express deterministically the behavior of the system, since they are a set of differential equations describing the evolution of the mean properties. Figure 6 shows the relationship between the results of the MEs and the result of the REs for a given observed random variable. REs are a fluid approximation of the system. As the time goes to infinity, MEs converge to the steady state distribution if it exists, otherwise yield the transient over any given interval. REs converge to the mean value if steady state exists, otherwise they are meaningless.

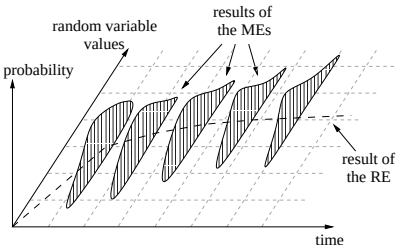


Fig. 6. Results of the Master Equations and the Rate Equations

The methodology we propose provides the solution for the MEs, and hence the complete system characterization.

IV. Monte Carlo Integration of the Master Equations

The set of MEs that describe the distribution process cannot in general be solved in closed form. However, the structure of the transition matrix that describes the stochastic process is very well suited for an efficient numerical solution based on Monte Carlo techniques [22][23][24][2], i.e., for a solution based on process simulation.

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 following the graph building rules given in Sect. II and the same rules define a transition matrix with good local properties, i.e., given a state there are few states where the process can evolve and, from the reward point of view, they are similar one another, so that there are not "diverging paths" that may lead to instabilities in the solution.

Samples obtained via Monte Carlo techniques are i.i.d. by construction, so that confidence intervals can be estimated on the whole probability distribution.

The key strength of the methodology is not only its efficient numerical solution: indeed, this method provides great flexibility in the system description and specification. On the one hand, this is like a generic simulation approach, but, being based on formal definitions, avoids the risk of incomplete or bugged specifications; on the other hand, assumptions made in fluid models can be avoided, since we can describe the system behavior in full detail. The implementation of the Monte Carlo integration of the model presented in this paper is available at [26].

A. Comparison with Fluid Models

We consider a very simple case in order to show the differences between ME and RE based approaches. Consider the case where a node updates its indegree only during update events. We assume infinite upload and download bandwidths and no constraints on the maximum outdegree. If $d_i(t)$ is the indegree at time t, at every update event the node will add $R - d_i(t)$ parents. In fact, under these assumptions the probability to find all the necessary parents to obtain all the stripes is 1, since there is always a node that is able to provide a connection. The differential equation that describes the evolution can be written as

$$\frac{d}{dt}d_i(t) = \lambda_{\text{up}}(R - d_i(t)) - d_i(t)\mu \qquad (6)$$

The second term considers the fact that the $d_i(t)$ parents can leave with rate μ each. Actually, not only parents can leave, but any ancestor of the node may disappear, thus the rate μ should consider also this aspect. Since we are looking for a simple closed form solution to this example, we assume that each node is able to build any stripe k starting from the set of stripes it is

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

receiving. This assumption is unrealistic, but it simplifies the example since the failure of a node has impact only on children, not on the whole subtree. Eq.(6) describes the evolution of the indegree for this system. We have also modified our numerical solution including the same hypothesis in order to compare the results.

Considering the initial condition $d_i(0) = 1$ (we suppose that all nodes are present at the beginning with exactly one parent each) the solution of (6) is

$$d_i(t) = \frac{\lambda_{\text{up}} R}{\lambda_{\text{up}} + \mu} \left(1 - e^{-(\lambda_{\text{up}}+\mu)t}\right) + e^{-(\lambda_{\text{up}}+\mu)t} \quad (7)$$

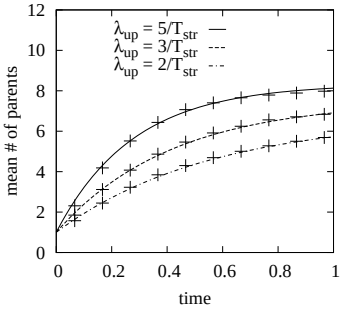
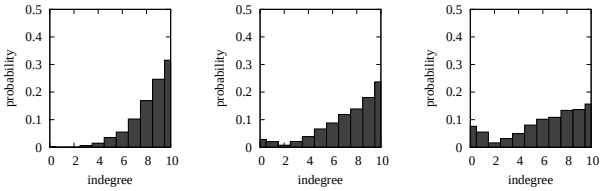


Fig. 7. Solution of the differential equation and the Rate Equation.

In Fig. 7 we compare the analytical solution of this very simple case with the solution of the Rate Equations (5) derived from our model. We set $R = 10$ stripes, $\mu = 1/T_{\text{str}}$ and $\lambda_{\text{up}} = \frac{5}{T_{\text{str}}}, \frac{3}{T_{\text{str}}}$ and $\frac{2}{T_{\text{str}}}$. We normalize the time with respect to T_{str}. The numerical solution follows closely the analytical one. But the results obtained from our model give more insight. In fact, we can observe how the *full* indegree distribution changes over time.

Fig 8 shows for instance the distribution of the number of parents (indegree) at time $T_{\text{str}}/2$ for different values of λ_{up}.

Notice that there is a non-null probability that a node remains without parent, thus being disconnected entirely from the distribution process, a phenomenon that a fluid approach analyzing the means entirely disregards, while in most cases it is one of the most important performance.



(a) $\lambda_{\text{up}} = \frac{5}{T_{\text{str}}}$ (b) $\lambda_{\text{up}} = \frac{3}{T_{\text{str}}}$ (c) $\lambda_{\text{up}} = \frac{2}{T_{\text{str}}}$

Fig. 8. Indegree distribution at time $T_{\text{str}}/2$ obtained from the solution of the MEs.

V. Application of the Methodology

A. System Description

We use a configuration with $N = 10^4$ nodes, but we have also checked some configurations with 10^5 nodes obtaining similar results. We use the input bandwidth distribution reported in Table II; bandwidths are expressed as a multiple of the streaming rate r_{str}. The streaming rate is divided into R' stripes and the source generates R stripes. Results are obtained for $R = 12$ and $R' = 3, 6, 9$.

TABLE II
Upload bandwidth distributions (normalized w.r.t. r_{STR})

Bandwidth	% nodes
1	20%
2	40%
5	40%

We consider an observation time equal to T_{str} (stream length). We consider two arrival patterns, with an initial number of nodes equal to $\frac{N}{10}$ and $\frac{N}{2}$ respectively; the remaining nodes arrive within $T_{\text{str}}/5$. The mean sojourn time is set to $0.5T_{\text{str}}$, T_{str}, and $2T_{\text{str}}$.

Each node can have up to 60 neighbors in the overlay graph (the actual number of neighbors depends on dynamics of the nodes); among these relationships, while uploading a node can have a maximum outdegree that is limited only by its bandwidth[3].

The stream is chunk based (e.g., few video frames or a slice of a few tens of milliseconds of sound) and we normalize the dimension of the chunk, U, such that $\frac{U}{r_{\text{str}}} = 1$ unit. A node can upload the content after a delay equal to the download time of a single chunk. So the delay can be considered as the "distance" (relative delay) of the node from the source of the stream. The length of the stream, T_{str}, is set to $10000\frac{U}{r_{\text{str}}} = 10000$ units.

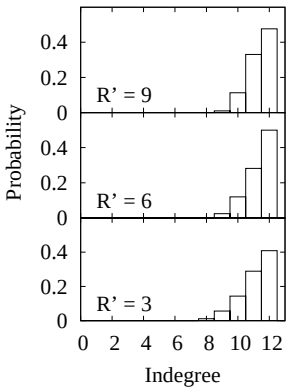
Besides degree and delay properties, we consider also the *quality of the mesh*: when a node remains orphan of an active parent, it switches to one of its standby parents: if they have enough bandwidth to help the node, the node has no service disruption; if no standby parent is able to help the node, it must search for a new parent, with a possible service disruption. We measure the quality of the mesh as the percentage of nodes that successfully switch to standby parents.

Due to space constraints, we report only some sample results that show the potentiality of the analytical framework. For an extended set of results refer to [25].

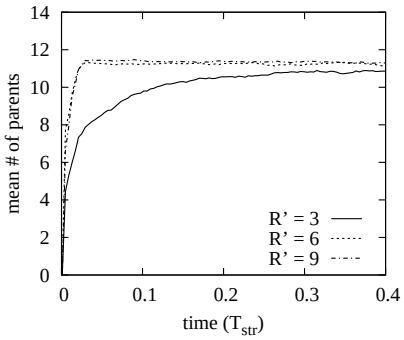
B. Analysis of the Indegree

Analyzing the indegree we examine whether the subdivision in stripes helps the distribution process or not. On the one hand, more stripes means that each stripe has a lower rate, so the loss of a single stripe has less impact. On the other hand, each node must maintain more active connections and the probability that any one of these connections fails increases.

Figure 9(a) shows the indegree distribution of the nodes at time $t = T_{\text{str}}$, computed with Eq. (3). In this case we have an initial number of nodes equal to $N/10$ and mean sojourn time T_{str}. The distribution tends to peak around R independently from R'. This means that all the nodes in the network are able to receive the full quality, since the degree is always greater or equal to R'. Note that with $R' = 9$ there is a fraction of the nodes with exactly 9 parents: this means that, in case of one parent that has left, the quality received by the node may be temporarily affected. For smaller values of R' all nodes always receive at least one redundant stripe, which makes them less vulnerable to disruptions.



(a) Prob. Distr. at T_{str}



(b) Evolution in time

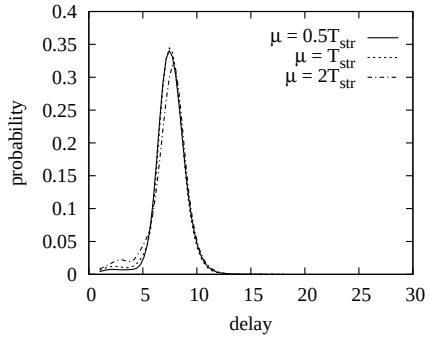
Fig. 9. Solution of the MEs for the indegree (initial number of nodes: $N/10$; sojourn time: T_{str}).

[3]The bandwidth is not necessarily the physical bandwidth, but can be the amount of resources willingly shared with the stream.

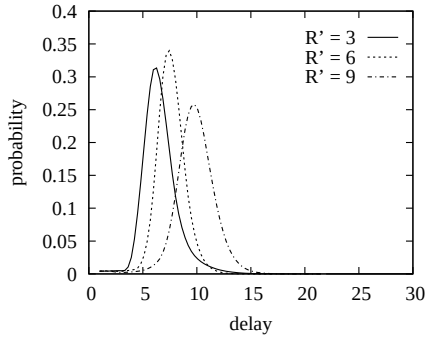
The temporal behavior of the indegree can be analyzed looking at the results of the rate equations (Fig. 9(b)) computed with Eq. (5). A stable value is reached quickly with the only exception of $R' = 3$: this means that the structure, even in presence of high churn is able to maintain a high quality of the stream.

C. Analysis of the Delay

The delay, expressed as time units, represents the number of hops from the source. We plot the probability density function of the delay. We consider $R' = 6$ and we set different sojourn times (μ). Fig. 10(a) shows the case of initial number of nodes equal to $N/2$. The distribution is not affected by the different values of μ. Similar results are obtained with the other configurations.



(a) $R' = 6$



(b) Different R'

Fig. 10. Distribution of the delay (initial number of nodes: $N/2$).

In Fig. 10(b) we show the impact of R' on the delay. Increasing the number of stripes has a price: since each node needs all the R' stripes to correctly play the stream, the absolute delay is given by maximum delay among the stripes. By increasing the number of stripes, the probability to have higher delays increases, since we have to compute the maximum among an increased number of stripes.

D. Analysis of the Quality

Aggregate results for the indegree and the delay are not able to capture all the aspects related to the quality of the received stream by a generic node i. In Table III we summarize other results that can be obtained from the solution of the MEs. The value of the *churn* is computed according to the arrival pattern: arrivals and departures are Poisson processes with rate $\lambda(t)$ and $\mu(t)$ respectively, so we can calculate the cumulative number nodes that have left at time T_{str} and consequently the value of churn.

TABLE III
OTHER STATISTICS.

R'	$1/\mu$	% Churn	%Switch
6	$0.5T_{\text{str}}$	186.8%	99.6%
6	T_{str}	93.5%	99.7%
6	$2T_{\text{str}}$	46.8%	99.8%
9	$0.5T_{\text{str}}$	187.3%	94.7%
9	T_{str}	93.2%	97.9%
9	$2T_{\text{str}}$	46.7%	99.1%

One of the performance index monitored is the probability to switch to a standby parent if an active parent leaves. This is given by $p(k_i, t)$ with $k_i < R'$. Integrating over time t we are able to compute the switch probability (see last column of Table III). With a small R', the percentage of switches is very close to 1, i.e., the received stream is stable. On the other hand, with R'

near to R, with high churn, if the number of parents of a node n drops below R', the probability to switch to a standby parent is 94%. This means that the quality temporarily decreases, as expected looking at degree distribution (Fig. 9(a)).

VI. Comparison with Simulations

In order to validate our analytic model and our assumptions — mainly exponential distributed times — we implement a simple overlay streaming protocol on top of the PeerSim P2P network simulator [27]. PeerSim 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 PeerSim simulator the interested reader is referred to [27].

A. Protocol Description and Simulation Set Up

We implemented the overlay streaming protocol using the event-driven engine. The protocol does not contain all the features of a real system, but it captures the essential behavior of the management of the distribution structure. The implementation is available at [28].

In the following we give a high level view of the protocol messages, leaving out details about the management of all the situations. The basic control messages exchanged by nodes are:

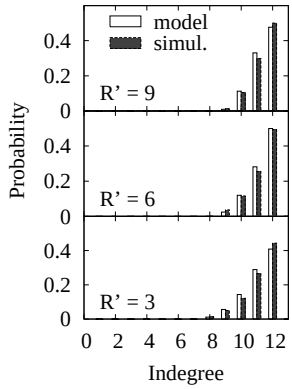
- *Join*: when a node joins the network, it obtains a list of neighbors from a *rendevouz server* and it sends messages to a subset of the received list of neighbors asking for stripes; with this message, a node asks to its neighbors to attach to a stripe.
- *Leave*: when a node decides to leave, it informs its neighbors; the messages are sent out after a random interval (different for each neighbor). This behavior is equivalent to have neighbors that periodically send to the node ping messages in order to check if it is still online.
- *Switch*: when a node remains orphan of an active parent, it sends a message to its standby parents asking to switch the status.

After receiving message, a node processes it determining, for instance, the availability of the bandwidth or the delay from the source, and replies with a message containing the requested parameters. Besides these procedures, a node periodically schedules an *Update* where it sends *Join* messages in order to increase its connectivity.

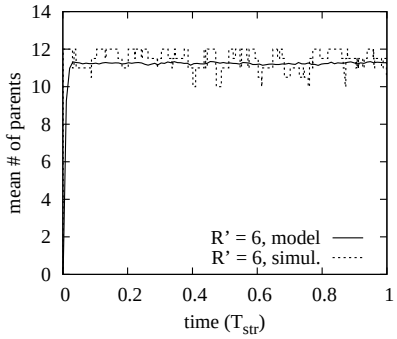
When a node joins the network, it selects a lifetime uniformly distributed between zero and twice the sojourn time used in the model, i.e. with a mean equal to the sojourn time used in the model. This distribution is used to check the impact of the hypothesis of exponentially distributed sojourn times we made in the model. At the transport layer, each message experiences an end-to-end delay that is uniformly distributed between a minimal and maximal value. The other constraints concerning the minimum and maximum number of children, number of stripes, or initial number of nodes are the same as in the model.

B. Simulation Results

We consider the impact of the number of stripes on the connectivity, i.e., how many stripes a node is able to receive when stripes have different sizes. Figure 11(a) shows the probability density function of the indegree at the end of the stream, compared with the results obtained with the model. We observe that results are very close, i.e., the impact of the assumption we made in the model is low.



(a) Prob. Distr. at T_{str}



(b) Evolution in time ($R' = 6$)

Fig. 11. Results obtained by simulation for the indegree (initial number of nodes: $N/10$).

Looking at the evolution in time of the outdegree, in Fig. 11(b) we compare the mean number of parents obtained from the model and the number of parents of a realization obtained from a simulation. As we can see, the model is able to closely predict the behavior of the system.

We then consider the distribution of the delays, i.e., the distance of the nodes from the source. Figure 12 shows the distribution of the delay for different values of R': continuous lines are the results predicted by the model, while dashed lines with points are the results of simulations. It is possible to see that the behavior of the system remains the same.

For all the experiments, we check also configurations

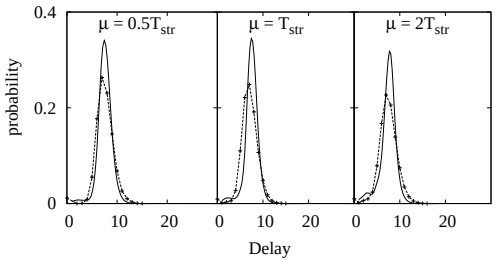


Fig. 12. Distribution of the delay obtained by simulation with PeerSim (initial number of nodes: $N/2$; $R' = 6$).

where we add a delay in the message transfer uniformly distributed between 0 and 1 (1 unit is the time necessary to transfer a single chunk). This delay (results not shown here) does not have an impact on the final performance. Moreover, we consider different scenarios where a node reacts in different ways in case of an active parent leaves and the node has no standby parents. In this case, the node, instead of waiting for the next *Update* event, can look for new parents. Results (not reported here, see [25] for details) shows that the distribution of the number of parents peaks around the maximum number of parents, and the overall quality increases. Nevertheless, the number of additional messages represents an overhead for the network. From the comparison between simulation and analytic results we can conclude that our analytic model is able to capture the essential performance characteristics of the overlay streaming systems.

VII. Discussion and Conclusions

The contribution of this paper is the introduction of a novel methodology for the high-level representation of overlay streaming systems. The proposed methodology does not represent only a model for a specific protocol. Actually, the general definition of the structure is flexible and can be adapted to a *generic* overlay streaming protocol that builds structured mesh.

Based on the use of Master Equations, the solution of the model yields the entire probability distribution and not only the mean, of the metrics of interest (node degree or delay) as well as the temporal (transient) dynamics.

We have modeled some systems proposed recently obtaining novel insights in the dynamics of self-organizing systems for streaming distribution. In the following we summarize the main findings that can help in designing better P2P streaming systems.

- Redundant stripes play a fundamental role in obtaining good performances. Recent proposals [5][10] consider only a small fraction of redundant information so, in case of node departures, the streaming is vulnerable to disruptions.
- The delay is influenced by stripe 'size': the greater R' (smaller stripes) the higher the delay. The number of necessary stripes R' should be kept low to keep a low delay. The delay remains low independently from the dynamics of the network.
- Under medium to high churn, nodes may experience a poor quality. Only stable nodes can prevent this behavior. This performance measure cannot be computed with any methodology that only yields averages.

The model can be extended in different ways in order to study different scenarios. For instance, we can consider *bandwidth fluctuations*, that model the unstable behavior of nodes. When the bandwidth decreases, a node simply drops some of the stripes. When the bandwidth increases, the node can accept new children when other nodes perform the update procedure. Another interesting extension is considering different policies for selecting the stripes from neighbors: instead of choosing randomly neighbor by neighbor, a node may collect all the stripes a set of nodes can give, selecting the combination of nodes that maximize the number of received stripes.

APPENDIX A
PROCEDURES

This appendix contains the formal definition of the procedures used to find the transitions between a state $\boldsymbol{S}$ and a new state $\boldsymbol{S'}$ in case of *Join* or *Update* event, as well as the procedure to find the delay of a node b given a connectivity matrix $\boldsymbol{S}$.

Join and Update

A node receives the vectors containing the stripes of its neighbors. The output of the decision process is a vector of R elements (R is the number of stripes), where element k contains the neighbor from which the node download stripe k (the first R' will be active, the remaining standby).

In order to find all the possible combinations of neighbors that can provide the stripes, we use as a basic building block the procedure that is able to find all the permutations of R objects taken from a set of B objects, where B is the number of neighbors. The number of available permutations is $(B)_R = B!/(B-R)!$, so the output is a matrix E with $(B)_R$ rows and R columns. Each element e_{ij} contains the neighbor that will provide stripe j in the combination i. If, looking at the vector provided by neighbor e_{ij}, we found a zero at position j, then the element e_{ij} is set to NULL. After this operation we obtain a new matrix M'.

The matrix M' is then reduced eliminating all the rows that are *equivalent* or *contained* in other rows. Row a is *equivalent* to row b if they have exactly the same non-null elements in the same positions. Row a is *contained* in row b if row b has the same non null elements of row a (in the same positions) and one or more other non null elements that row a does not have.

The final matrix M'' contains all the possible combinations neighbor-stripe that the node can select. Note that, with the Monte Carlo integration methodology we use, it is not necessary to compute the entire matrix M'' (it would be computationally expensive): since we perform a realization of the process, it is sufficient to generate a random row of the matrix for each realization[4].

Computing the Delay

In order to find the delay, in terms of number of steps from the root, we start from the connectivity matrix $\boldsymbol{S}$. Given a node i the procedure to find the delay is described in Algorithm 1. The procedure "find_stripes(stripe_id, column_index)" returns the row index where the stripe is. The source node has index zero, so when we reach the source the procedure stops. This procedure is done for all nodes except the source.

Algorithm 1 Procedure for finding the number of steps

```
input: connectivity matrix S, initial node i;
output: number of steps from root;
#steps = 0;
#parents = 0;
for α_k = α_1, α_2, ..., α_R do
    j = find_stripe(α_k, i);
    while j ≥ 0 do
        #parents++;
        i = j;
        j = find_stripe(α_k, i);
    end while
    #steps = max(#steps, #parents);
end for
return(#steps);
```

[4]Consider the equivalent problem of generating all the possible permutations of a set of B elements, compared to the cost of generating a random permutation.

REFERENCES

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

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

[3] AD. A. Tran, K. A. Hua, and T. T. Do, "A Peer-to-Peer Architecture for Media Streaming," in *IEEE JSAC: Special Issue on Advances in Overlay Networks*, Vol.22, N.1, Jan. 2004.

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

[5] X. Zhang, J. Liu, B. Li, and T. S. P. Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Live Media Streaming," in *Proc. IEEE INFOCOM 2005*, Miami, Mar. 2005.

[6] F. Pianese, J. Keller, and E. W. Biersack, "PULSE, a Flexible P2P Live Streaming System," in *Proc. 9th IEEE Global Internet Symposium 2006*, Barcelona, Spain, Apr. 2006.

[7] S. N. Dorogovtsev, and J. F. F. Mendes, "Evolution of Networks: From Biological Nets to the Internet and WWW," Oxford University Press, Oxford, January 2003.

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

[9] N. Magharei, R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-based Streaming," in *Proc. IEEE INFOCOM 2007*, Anchorage, Alaska, May 2007.

[10] Y. Sung, M. Bishop, and S. Rao, "Enabling Contribution Awareness in an Overlay Broadcasting System," in *Proc. SIGCOMM 2006*, Sept. 2006.

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

[12] G. Tan, S.A. Jarvis, X. Chen, D.P. Spooner, and G.R. Nudd, "Performance Analysis and Improvement of Overlay Construction for Peer-to-Peer Live Media Streaming," in *Proc. MASCOTS 05*, Sept. 2005, GA, USA.

[13] T. Small, B. Liang, and B. Li, "Scaling Laws and Tradeoffs of Peer-to-Peer Live Multimedia Streaming," in *Proc. ACM Multimedia 2006*, Santa Barbara, California, October 23-27, 2006.

[14] K. Andreev, B.M. Maggs, A. Meyerson, and R. Sitaraman, "Designing Overlay Multicast Networks for Streaming," in *Proc. SPAA*, June 2003.

[15] R. Kumar, Y. Liu, and K.W. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. IEEE INFOCOM 2007*, Anchorage, Alaska, May 2007.

[16] S. Tewari, and L. Kleinrock, "Analytical Model for BitTorrent-based Live Video Streaming," in *Proc. NIME 2007 Workshop*, Las Vegas, NV, Jan 2007.

[17] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," in *Proc. IEEE INFOCOM 2007*, Anchorage, Alaska, May 2007.

[18] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations," in *Proc. 11th ACM SIGKDD 2005*, Chicago, IL, USA, Aug. 2005.

[19] D. Carra, R. Lo Cigno, and E. W. Biersack, "Graph Based Modeling of P2P Streaming Systems," in *Proc. IFIP Networking 2007*, Atlanta, Georgia, USA, May 2007.

[20] V. K. Goyal, "Multiple Description Coding: Compression Meets the Network," in *IEEE Signal Processing Magazine*, pp. 7493, Sept. 2001.

[21] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: peer-to-peer media streaming using Collect-Cast," in *Proc. ACM Multimedia 2003*, Berkeley, CA, Aug. 2003.

[22] H. P. Breuer and F. Petruccione "On the numerical integration of Burgers' equation by stochastic simulation methods," *Computer Physics Communications*, Vol. 77, Pages 207-218, 1993.

[23] J. Honerkamp, "Stochastic Dynamical Systems: Concepts, Numerical Methods, Data Analysis," 1994, VCH, New York.

[24] D.T. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *Journal of Physical Chemistry*, Vol. 63, Issue 25, Pages 2340-2361, 1977.

[25] D. Carra, "Performance Evaluation of Overlay Content Distribution Systems," PhD Thesis, Univ. of Trento, March 2007.

[26] Stochastic Graph Process Solver, `http://dit.unitn.it/networking/tool/groover/`

[27] PeerSim: A Peer-to-Peer Simulator, `http://peersim.sourceforge.net/`

[28] Content Distribution Protocols in PeerSim, `http://dit.unitn.it/networking/tool/cdp-peersim`

Damiano Carra received his Laurea degree in Telecommunication Engineering from Politecnico di Milano in 2000 and his Ph.D. in Computer Science from 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 modeling and performance evaluation of peer-to-peer networks.

Renato Lo Cigno is Associate Professor at the Department of Computer Science and Telecommunications (DIT) of the University of Trento, Italy, where he is one of the founding members of the Networking research group. He received a Dr. Ing. degree in Electronic Engeneering from Politecnico di Torino in 1988. From 1999 to 2002 was been with the Telecommunication Research Group of the Electronics Department of Politecnico di Torino. From June 1998 to February 1999, he was at the CS Department at UCLA as Visiting Scholar under grant CNR 203.15.8. He is coauthor of more than 100 journal and conference papers in the area of communication networks and systems. His curent research interests are in 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. Renato Lo Cigno is member of the IEEE and ACM; he is Area Editor of Computer Networks (Elsevier), and has served as Chair os TPC members of several leading conferences, including IEEE INFOCOM, IEEE Globecom, IEEE/ACM MSWiM, IEEE MASS, ACM WMASH.

Ernst Biersack received his M.S. and Ph.D. degrees in Computer Science from the Technische Universität München, Munich, Germany and his habilitation from the University of Nice, France.

Since March 1992 he has been a Professor in Telecommunications at Institut Eurecom, in Sophia Antipolis, France. His current research is on Peer-to-Peer Systems and Network Tomography.