Multistage Congestion Games for Live-Streaming

Gabriele D'Angelo
Department of Computer Science
University of Bologna
Bologna, Italy
gdangelo@cs.unibo.it

Stefano Ferretti
Department of Computer Science
University of Bologna
Bologna, Italy
sferrett@cs.unibo.it

Giovanni Rossi
Department of Computer Science
University of Bologna
Bologna, Italy
giorossi@cs.unibo.it

***Abstract*—In this paper we model peer-to-peer real-time streaming by resorting to multistage congestion games. Based on this, we identify a set of strategy profiles through which the stream may be responsively distributed to all peers. We then provide strategy restriction mechanisms which allow to obtain equilibria where both streaming duration and congestion are minimized. From this modeling, a distributed algorithm is proposed (ConGaS), which can be easily executed at peers, to let them coordinate to optimize the streaming. An experimental evaluation is performed to compare the results obtained by ConGaS against two other dissemination strategies. Simulation results confirm the viability and the efficacy of this proposal.**

I. Introduction

Today, real-time streaming is considered one of the most interesting applications in distributed systems, because of the great interest shown by customers, the consequent attention paid by software companies, and also due to the raised technical challenges to factually develop them, which attract more and more researchers. Essentially, the problem is concerned with those many situations where some media content is produced and distributed in time. One *new* content unit is periodically generated by a source (or broadcaster) and made available to a set of interested users. The main requirement here is that these content units must be timely disseminated to all users which are interested in. In other words, such distribution should guarantee that the delay between the generation of a novel media content, and its reception at a given user is kept within a limited amount of time, with a small variance of such delays for different content units.

While actual commercial solutions resort to classic, non-scalable centralized approaches where the broadcaster acts as a media server that dispatches all produced contents to all users, another interesting and promising architectural solution is to employ a peer-to-peer scheme, which allows peers to share their possessed content units, in order to fasten the content distribution process [1], [2], [3], [4], [5], [6]. Streaming peer-to-peer approaches address the problem of disseminating data with the creation of specifically designed overlay networks. An overlay network can be designed in different ways. Peers can be logically organized as a tree, where the data flows from the root, i.e. the broadcaster, to the leaves [7], [8], [9], a multi-tree [6], [10], or a generic mesh [11], [12], [13]. Alternatively, more sophisticated approaches can be utilized that, for instance, exploit DHTs for the content distribution, e.g. [4], [14]. In any case, when a distributed system is going to be developed for the support of live streaming applications, three main concerns to consider are: *i)* the overall load due to forwarding activity should be evenly shared among the participants; *ii)* the time needed for full dissemination should be minimized; and *iii)* the protocol should be fair in that the expected time needed to receive the whole content should be the same across all peers. In this respect, in recent times game theory is proving very useful for modeling communication systems and dynamic distributed environments. Results coming from these studies allow to identify technical solutions that take into consideration all three concerns mentioned above.

Game theory results have already been used to design effective mechanisms for live content distribution, but yet it turns out that only a limited amount of proposals is available. Among these, the majority refers to payment schemes and incentives-based approaches [1], [15], [16], [17], [18], [19]. Some works focus on routing problems. Often, this issue is addressed in terms of congestion (or, more generally, potential) games [20]. In this paper, we model real-time streaming scenarios in terms of multistage congestion games. Specifically, we consider a peer-to-peer approach. The cooperation among peers is not accomplished by resorting to some structured approach, as in other proposals. Rather, peers coordinate by means of a protocol based on multistage congestion games. Our modeling allows to distinguish between the number of stages needed to fully disseminate the whole content (or streaming length) on the one side, and stage-wise congestion on the other. In addition, and most importantly, it enables to identify a strategy restriction mechanism which at each stage prevents peers from asking certain content units, given the prevailing content distribution over the population. With such restrictions, at equilibrium both streaming length and stage-wise congestion are minimized. This is detailed by means of a distributed algorithm *ConGaS* (i.e. *Con*gestion *Ga*mes for *S*treaming) that we develop for implementing the proposed equilibrium selection method. We compare the proposed approach with respect to a mechanism that implements less restrictions for the peers' interactions. Moreover, to obtain insightful results, ConGaS is compared to a basic gossip protocol for content dissemination.

The paper is organized as follows: in Section II we introduce the theory based on the multistage congestion games to model live streaming applications. In Section III, we show that a

Giovanni Rossi, Gabriele D'Angelo, Stefano Ferretti
Multistage Congestion Games for Live-Streaming
Proceedings of 15th IEEE Symposium on Computers and Communications (ISCC 2010)
Riccione, Italy, June 2010

simple strategy restriction mechanism allows to minimize, at equilibrium, both streaming length and stage-wise congestion. In Section IV the model is translated into a distributed algorithm; its functioning is evaluated in Section V. Section VI provides insights about the implementation issues of ConGaS. Some remarks conclude the paper (Section VII).

II. Modelling Live Streaming with Multistage Congestion Games

A. Background

We employ a round-based time modeling. Stream production occurs over a finite time-sequence $t = 0, 1, \ldots, T$. The *source* (denoted by 0) provides one new unit c^t of content at each stage/round t. A set $N = \{1, \ldots, n\}$ of *peers* is involved in content distribution. Also let $N_0 = \{0, 1, \ldots, n\}$.

Stream distribution may be modeled as a *multistage game* by means of *congestion game forms* [21]. A multistage game can be viewed as a *tree*, whose nodes correspond to a moment at which at least one player has to take action. Paths in this tree correspond to distinct courses the game may take. In the model we assume *perfect information*: when asked to take action, at any time t, all players know exactly what node has been reached at t (then we will show how this can be factually implemented in the distributed algorithm). A *strategy*, for a player, specifies an (admissible) action to take at each node. In our game, players are the peers.

When modeling real-time streaming as multistage congestion game, it is necessary to define the *rules* of the game (or constraints on the players' actions), which in this case are

Ru1: at each round t the source can send each unit c^t only to one peer;

Ru2: if a peer at t has received units $c^{t_1}, \ldots c^{t_h}$, $0 \leq t_1, \ldots, t_h < t$, then at each round this peer can send only one of such units and to one other peer;

Ru3: in any round, each peer may receive some unit either from one other peer, or else from the source.

Nodes in the game tree are identified each by a time-indexed set $C^t = \{C_0^t, C_1^t, \ldots, C_n^t\}$, where $C_i^t \subseteq \{c^0, c^1, \ldots, c^t\}$ specifies what content units peer $i \in N_0$ has received at time t. For every $i \in N$, a strategy specifies, for each game tree node C^t, some $j \in N_0$ from whom to ask, in round t, a content unit $c^{t'}, 0 \leq t' \leq t$. These strategies are finite sequences as long as some upper bound T_* on duration exists.

Let $\mathcal{G}$ denote the set of all game tree nodes, i.e. the family of all possible content distributions over peers that may be reached along some game course, under Ru1-3, but independently from what game courses prevail, and at what time. In our model, a strategy A^i for peer $i \in N$ has form $A^i : \mathcal{G} \to N_0$, with $A^i(C) = j \neq i$ denoting the one $j \in N_0$ from whom i asks to receive at game tree node $C \in \mathcal{G}$. In the following, we use players' strategies (and restrictions on them) to identify a viable algorithm for real-time streaming.

B. Congestion Games and Forms

In a congestion game form there is a set N of players and a set M of facilities, and each player $i \in N$ has a set $\Sigma^i \subseteq 2^M$ of strategies, where 2^M is the (power) set of all subsets of M. Usually, M is the edge set of a graph, and each player $i \in N$ has to reach a destination v_i^d from an origin v_i^o. Then, the set Σ^i of strategies for i contains all (edges of) $v_i^o - v_i^d$-paths.

A *congestion game form* $F = (N, M, \Sigma^1 \times \cdots \times \Sigma^n)$ identifies a whole class of congestion games, each obtained by specifying the payoffs $\pi^i : \Sigma \to \mathbb{R}_+$ of players $i \in N$, where $\Sigma = \Sigma^1 \times \cdots \times \Sigma^n$. Profile $A = \{A^1, \ldots, A^n\} \in \Sigma$ of strategies identifies *congestion vector* $\sigma(A) = \{\sigma_a(A) : a \in M\}$ specifying how many players have each facility $a \in M$ in their strategy A^i. That is, $\sigma_a(A) = |\{i \in N : a \in A^i\}|$. The game is *monotone* when each $a \in M$ has an associated utility function $u_a : \mathbb{Z}_+ \to \mathbb{R}_+$ satisfying $u_a(k) < u_a(k')$ whenever $k > k'$, and each $i \in N$ gets a payoff given by the sum over all the chosen facilities $a \in A^i$ of the corresponding utility: $\pi^i(A) = \sum_{a \in A^i} u_a(\sigma_a(A))$. Finally, a congestion game form (and any game derived from it) is *symmetric* when the strategy set is the same across players: $\Sigma^1 = \cdots = \Sigma^n$ [21].

P2P streaming systems may be approached through congestion games with facilities being players themselves: every strategy profile $A = (A^1, \ldots, A^n)$ has an associated *congestion matrix* $\sigma(A) = \{\sigma_C^i(A) : C \in \mathcal{G}, i \in N_0\}$, where

$$\sigma_C^i(A) = |\{j \in N : i = A^j(C), C_i \not\subseteq C_j\}|$$

is the number of peers who ask to receive from $i \in N_0$ some content that *this latter has but they miss* (at generic game tree node $C \in \mathcal{G}$) [1].

Denote by $\kappa = |\mathcal{G}|$ the whole number of game tree nodes. A strategy A^i for a peer $i \in N$ can be regarded as a point $A^i \in N_0^\kappa$, as it specifies somebody (although possibly with no additional content) to ask from at each node $C \in \mathcal{G}$ that may be reached. Hence, the corresponding congestion game form is $F = \left(N, N_0^\kappa, \underbrace{N_0^\kappa \times \cdots \times N_0^\kappa}_{n}\right)$.

Players' payoffs $\pi^i : N_0^{\kappa n} \to \mathbb{R}_+$ are assumed to consist of a sum over nodes of some utility or *per-node* payoff received at each game tree node $C \in \mathcal{G}$. This utility depends on the prevailing content distribution (which is precisely what the game tree node C specifies), and on the profile $A^1(C), \ldots, A^n(C)$ of *per-node* strategies that players choose at node C. Hence, per-node payoffs received by peers $i \in N$ may depend on congestion, which here is the number of other peers $i' \in N$ with the same (valid) per-node strategy $A^{i'}(C) = A^i(C)$. This models real-time streaming in terms of congestion games with facilities being pairs (j, C), where $j \in N_0$ is either a peer or the source and $C = C^t$ is a game tree node or content distribution that may prevail at time t.

In a simplest form, the payoff $\pi^i(A)$ of any given strategy profile A to a peer i is the sum, over all conceivable game

[1] If a peer asks to receive from someone who has no additional content, then such a request is simply ignored by the system: it causes null congestion. A request is *valid* if it contributes to congestion.

tree nodes C, of the values taken by utility u_{C_j}, which in turn depends only on congestion $\sigma^j_C(A)$, that is,

$$\pi^i(A) = \sum_{C\in\mathcal{G}} \sum_{\substack{j\in N_0: C_j\not\subseteq C_i \\ A^i(C)=j}} u_{C_j}\left(\sigma^j_C(A)\right). \qquad (1)$$

For any strategy profile A, each peer i gets a utility at each node C which depends exclusively on the number $\sigma^j_C(A)$ of those with the same (valid) per-node strategy.

A profile $A = (A^1, \ldots, A^n)$ is Pareto-optimal if there is no profile $B = (B^1, \ldots, B^n)$ such that $\pi^i(B) \geq \pi^i(A)$ for all $i \in N$, with strict inequality for at least one i. Hence, from an aggregate perspective, Pareto-optimal profiles are efficient: there is no chance of improving someone's payoff without deteriorating someone else's one. Congestion games allow for neat conditions under which desirable properties, such as Pareto-optimality and strength of equilibrium, attain. In fact, in symmetric and monotone such games, these properties depend on the structure of the union $\Sigma^U = \cup_{i\in N}\Sigma^i$ of strategy spaces. In particular, on whether a *bad configuration* appears or not. Formally, Σ^U displays a bad configuration when there are three strategies $X, Y, Z \in \Sigma^U$ and two facilities $x, y \in M$ such that $x \in X \not\ni y$ and $x \notin Y \ni y$ but $x \in Z \ni y$. *Thus, two facilities give rise to a bad configuration if there are strategies in Σ^U which use one of them but not the other, and there is also a strategy in Σ^U which uses both of them. The latter never occurs if Σ^U consists of singletons* [21, pp. 87-88]. As the name suggests, it is desirable that no bad configuration exists. Here facilities are players themselves, although any fixed player corresponds to two distinct facilities when referring to two distinct game tree nodes. By Ru3 above, strategies are time-sequences of singletons, and hence the safe case applies.

If $\pi^i(A^{-i}, A^i) \geq \pi^i(A^{-i}, B^i)$ for all $i \in N$ and $B^i \in N_0^\kappa$, where $A^{-i} \in N_0^{\kappa(n-1)}$ is a $n-1$ profile for peers $j \in N\backslash i$ as well as $A^i \in N_0^\kappa$ is a strategy for peer i, then (A^{-i}, A^i) is an equilibrium. In particular, $A = (A^1, \ldots, A^n)$ is a *strong* equilibrium if for no coalition $\emptyset \neq S \subseteq N$ is there a choice of $B^i \in N_0^\kappa$ for coalition members $i \in S$ such that $\pi^i(B^S, A^S) > \pi^i(A)$ for all coalition members $i \in S$, where (B^S, A^S) denotes the profile in which each $i \in S$ chooses B^i and each $j \in S^c = N\backslash S$ chooses A^j. In words, no non-empty coalition can deviate from strong equilibrium profiles and thereby strictly increase the payoffs of *all* its members.

When considering the implications of strong equilibrium for $S = N$, one gets similar conditions as those identifying Pareto-optimal profiles. In fact, as strategies are time-sequences of singletons, the model provided thus far yields a symmetric monotone congestion game with no bad configuration, where therefore the set of strong equilibria is non-empty, coincides with the set of equilibria and, generically, is (weakly) included in the set of Pareto-optimal profiles [21].

C. Best-case Equilibrium

One indicator of streaming efficiency is simply the number of rounds needed to spread the whole content over the whole peer set N. Assume the number of peers is a power of 2, that is, $n = 2^m$ for some natural m. Under our assumptions Ru1-3, any content unit can spread over the whole population no faster than through $m+1$ (consecutive) rounds. This is easy to see, as each peer can send/receive a content unit per round, while the source is allowed to distribute that unit only once, to a single peer. Therefore, after the source transmits the unit to a given peer (this costs one round), the number of peers that possess that unit can at most double each round, hence the lower bound for a single unit distribution is $\log_2 n = m$ rounds. A crucial fact is that *all* the $T+1$ content units of a stream may be distributed to all peers in exactly $m+1$ rounds. Still, in view of Ru1-3 this can only be achieved if whenever a peer receives a unit c^t in round $t+k$, this peer forwards c^t to other peers for the remaining $m-k$ rounds. As a consequence, being already involved in the distribution of c^t, during these rounds this peer cannot receive units to be further forwarded. That is to say, during each of these remaining $m-k$ rounds, the peer can receive only some unit that completes its distribution at that round, i.e. the peer is among those 2^{m-1} who are the last ones to receive that unit. This not only is feasible, but can be obtained through many different streaming trees, as detailed in the following.

Definition 1: profile $A \in N_0^{\kappa n}$ is *deterministic* if $\forall i \in N$,

(a) $\left|\left\{C \in \mathcal{G} : A^i(C) = j, C_j \not\subseteq C_i\right\}\right| = T+1$,
(b) $A^i(C) = j, C_j \not\subseteq C_i \Rightarrow |C_j\backslash C_i| = 1$ for all $C \in \mathcal{G}$.

Hence, each peer makes exactly $T+1$ valid requests to receive, and receives some (distinct) content unit for every request. Also, the $T+1$ valid requests made by any peer i are all addressed, each at a different game tree node, to someone who at that node has precisely one additional content unit. The name *deterministic* is due to the assumption that transitions from one game tree t-node C^t to $t+1$-nodes C^{t+1} may be stochastic: a generic strategy profile A does not yield a unique game course, but a probability distribution over game courses. Whatever its form, an underlying probabilistic model essentially decides who gets what when multiple peers ask to receive from a common j. Deterministic profiles actually allow to ignore the underlying probabilistic model, putting probability 1 on one game course and probability 0 on all other courses. Accordingly, consider the unique content distribution over peers reached at t by deterministic profile A, and denote it by $C^t(A) = \{C^t_0(A), C^t_1(A), \ldots, C^t_n(A)\}$.

Definition 2: a deterministic profile $A \in N_0^{\kappa n}$ is *fastest streaming* if for all $0 < k \leq t \leq T_*$,
$\left|\left\{i \in N : c^{t-k} \in C^t_i(A)\right\}\right| = \min\{2^m, 2^{k-1}\}$.

We denote by $\mathcal{A}^*$ the set of fastest streaming profiles. These profiles spread each content unit c^t over $2^0 = 1$ peer in round t (i.e. from the source to a peer), over (new) $2^0 = 1$ peer in round $t+1$ (i.e. from that peer to another), over $2^1 = 2$ peers in round $t+2$ (i.e. from the two peers that have the content to other two), and so on, until (new and final) 2^{m-1} peers receive unit c^t in round $t+m$, which is the $m+1$-th (i.e. final) round where this unit circulates.

Consider a generic t such that $m \leq t \leq T$. For any $A \in \mathcal{A}^*$, in round t there are exactly $m+1$ content units $c^t, c^{t-1}, \ldots, c^{t-m}$ being distributed across the whole population, out of which precisely m (i.e. $c^{t-1}, c^{t-2}, \ldots, c^{t-m}$) are sent by some peers to some other peers, while one unit (i.e. c^t) is sent from the source to some suitably chosen peer. Hence, in this round t each peer is a receiver (of some unit $c^{t-k}, 0 \leq k \leq m$). Conversely, only $2^m - 1$ peers also send (units $c^{t-k}, 1 \leq k \leq m$), as the source forwards c^t.

Definitions 1 and 2 may be turned into a useful method for establishing, for any course of the game reached up to any time $t \geq 0$, how to proceed in round t in order to have a streaming inducted by a strategy profile $A \in \mathcal{A}^*$. In fact, all priorities can be captured by the cited main constraint: for any $t \geq 0$, if in the previous round a peer has received and/or forwarded some unit that will have to be forwarded in round $t+1$ as well, then in this round t this peer cannot receive any unit that will also have to be forwarded in round $t+1$. Put it formally, let S_t^k be the subset of peers who *send* unit c^{t-k} in round t ($1 \leq k \leq m$), and R_t^k be the subset of peers who *receive* unit c^{t-k} in round t ($0 \leq k \leq m$). Then, ***if*** $i \in S_{t-1}^k$ *or* $i \in R_{t-1}^k$ for some $k \leq m-2$, *or* $t \geq m$ *and* $i \notin S_{t-1}^k$ for all $k \geq 1$, ***then*** $i \notin R_t^k$ for all $k \leq m-1$.

Many different streaming tree evolutions allow to spread the whole content over the whole peer population in a way such that each unit c^t reaches *new* 2^{k-1} peers in each round $t+k$ for $k = 1, \ldots, m$. Also note that all of them satisfy condition (a) (in Definition 1), as at any node C and for any two peers $i, j \in N$ we have $|C_i \backslash C_j| \in \{0, 1\}$. Still, with payoffs given by (1) above, fastest streaming is not sustainable at equilibrium, because condition (b) (in Definition 1) is too demanding: selfish (and myopic) peers try to receive some unit in any round until they get the whole content.

D. Worst-case Equilibrium

Given the P2P setting, where peers always satisfy precisely one (randomly selected) valid request among those received, in each round the number of distributed units equals the number of those who are asked to forward through some valid request. At equilibrium such a number equals the minimum between the number of those who have some units that someone else is missing and the number of those who miss some unit.

Claim: the upper bound for equilibrium streaming length is $T_* = T + 2^m + 1$. For space reasons the proof is omitted but can be found in [22].

Although worst-case equilibrium streaming length is linear in both the whole number of produced units and the whole number of peers, it can be rather greater than the optimal streaming length and not feasible for live-streaming applications.

III. Strategy Restriction

Within the proposed setting, we now provide strategy restrictions of the form: certain peers $i \in N$ at certain nodes $C \in \mathcal{G}$ cannot ask to receive from certain $j \in N_0$. In this way, the model defines a monotone congestion game with no bad configuration, where equilibria are strong and Pareto-optimal.

It is possible that at an equilibrium strategy profile A each content unit $c^t, 0 \leq t \leq T$ reaches 2^{k-1} *new* peers in each round $t+k, k = 1, \ldots, m$, and thus the whole population in $m+1$ rounds, which is optimal in terms of streaming length. Yet, such a profile A cannot be deterministic, as any equilibrium profile A must result in a congestion $\sigma_C^j(A) > 1$ for some (j, C)-entries of the associated matrix $\sigma(A)$. In fact, each (greedy) peer i makes a valid request to receive as long as the whole content has not been received. Hence, equilibrium profiles A surely yield some congestion, and provide a streaming length which ranges from the optimal (i.e. minimum) one $T+m+1$ to the worst-case one $T+2^m+1$. Then, for a social planner there are two priorities when designing restrictions: *i)* at equilibrium (with restrictions) streaming length should be $T+m+1$, the same as with fastest streaming profiles; *ii)* congestion should be minimized.

Consider a strategy restriction mechanism which specifies from what $j \in N_0$ each peer $i \in N$ can ask for content at each node C^t. In other terms, the mechanism specifies for any node $C^t = (C_0^t, C_1^t, \ldots, C_n^t)$ and for any $i \in N, j \in N_0$ such that $C_j \not\subseteq C_i$, whether it may be $j = A^i(C)$ or not. Consider the following per-node restriction mechanism: for all $i \in N$ and $C^t \in \mathcal{G}$

Rm1: if $c^{t-k} \in C_i^t$ for some $k \leq m-1$, then $A^i(C^t) \neq j$ for all $j \in N_0$ such that $c^{t-k'} \in C_j^t$ for some $k' < m$;

Rm2: if $c^{t-m} \notin C_i^t$, then $A^i(C^t) = j$ for some $j \in N_0$ such that $c^{t-m} \in C_j^t$.

In this way, if, given previous history, a peer in t has some content unit c^{t-k} that must be forwarded in round $t+1$, then in this round t the peer cannot ask to receive from those $j \in N_0$ who in t have units $c^{t-k'}$ to be also forwarded in round $t+1$ (i.e. such that $t - k' + m > t$).

Restriction mechanism Rm1-2 above is useful for exploiting selfish behavior toward socially desirable outcomes. In particular, the mechanism is simple and, most importantly, specifies conditions only in terms of the generic node C^t that may be reached at some time t during game course. More details can be found in [22].

IV. ConGaS

We now detail a distributed algorithm derived from the use of *Con*gestion *Ga*mes for *S*treaming as described thus far, and hence referred to as ConGaS. It provides minimal streaming length. For simplicity of exposition, it is here detailed for the case where the number of peers is a power of two. In fact, the algorithm works for any numbers of peers, although the general case requires additional procedures that can be ignored under our assumption $n = 2^m$.

A first point worth of mention is that in the theoretical model provided thus far, peers have a perfect knowledge of the game course. In other words, all peers know what other peers are doing, i.e. which contents they are sending/receiving during each round. Certainly, the use of communication protocols to

Algorithm 1 ConGaS

INITIALIZATION

```
1: p = IDNODE()
2: if (p == 0) then
3:     s ← GENERATESEED()
4:     BROADCAST(s)
5: end if
```

DISTRIBUTION LOOP

```
1: for all t ∈ [0, T] do
2:     NextFree ← N = N_0 \ {0}
3:     S^t = 0
4:     R^t = N
5:     for k = max(t − m, 0) to t do
6:         for all p ∈ S^k do
7:             MANAGEDISTRIBUTION(p, k)
8:         end for
9:     end for
10:    p = IDNODE()
11:    if (p == 0) then
12:        c^t = NEWCHUNK()
13:    end if
14:    MANAGEDISTRIBUTION(0, t)
15: end for
```

```
1: function MANAGEDISTRIBUTION(p, t)
2: recv ← NEXTRECV(R^t, s)
3: MUSTSEND(p, recv, c^t)
4: ADD(recv, S^t)
5: REMOVE(recv, R^t)
6: REMOVE(recv, NextFree)
```

```
1: function NEXTRECV(AvailRecvs, seed)
2: repeat
3:     n ← RANDOMPEER(seed)
4: until n ∈ NextFree ∩ AvailRecvs
5: return n
```

implement this sharing of global knowledge among peers is a non-viable solution. The time and bandwidth used to distribute all such information would extremely complicate the task of disseminating contents. Another, viable solution is to let peers share a seed employed to randomly generate same sequences of pseudo-random numbers. Such shared seed serves as the needed coordination mean among nodes (i.e. it is employed to randomly select those peers that receive any given content unit). A description of ConGaS follows (see also the reported Algorithm): during the initialization, the broadcaster (i.e. node 0) sends to all peers a generated seed value. In practice, this simple broadcast is the sole operation required to enable peers to have perfect knowledge on the distribution process. The distribution loop consists of an iterative behavior: each iteration t corresponds to the production, at the broadcaster, of a novel content unit c^t to be distributed. Meanwhile, on-going content units $c^{t-k}, k = 1, \ldots, m$ that have not yet been delivered to all peers are disseminated according to our method (lines 5-9 of the distribution loop). The MANAGEDISTRIBUTION() procedure, executed by all nodes, defines who sends what content unit to who. Differences in peers' *actions* are simply determined accordingly to their *id*s.

At each iteration t, each peer can receive a single, new content unit. This is achieved by picking nodes from an auxiliary list *NextFree*, which is initialized to N (line 2), and then progressively emptied through different calls of MANAGEDISTRIBUTION() (line 7). Specifically, given a content unit c^k being distributed ($t - m \leq k \leq t$), a bijection between those who have the unit (senders S^k in the code) and (some of) those who do not (receivers R^k in the code) is provided. Thus, for any content unit, at each h-th step of distribution, 2^h nodes have the content unit and 2^h are selected to be the receivers. Once the distribution for the on-going content units is specified, a new content unit is produced at the broadcaster (lines 11-13) and MANAGEDISTRIBUTION() is called for the novel content unit. In detail, in MANAGEDISTRIBUTION() a new receiver *recv* is identified thought NEXTRECV(). MUSTSEND() schedules the delivery of the content unit from sender *p* to *recv*. Moreover, *recv* is added into S^k (the list of c^k senders; line 4). Accordingly, *recv* is removed from the list of receivers R^k (as well as from the list of those who may be selected as receivers of ongoing content units; lines 5-6). NEXTRECV() randomly selects receivers. In simple words, a (novel) node is picked until someone is found in the intersection of *NextFree* and *AvailRecvs* (i.e. the set of those who have not already been selected as receivers of the considered unit). Finally, MUSTSEND() makes thus far identified senders actually send the on-going units, based on the nodes' *id*.

V. PERFORMANCE EVALUATION

The performance evaluation of ConGaS has been conducted following a simulation-based approach. In detail, a dedicated simulator has been designed and implemented to verify the validity of our assumptions and theoretical results. The simulator models the evolution of the distributed system both in terms of participants and communication protocols.

In order to evaluate our streaming method, ConGaS, it is necessary to find a metric and a benchmark for comparison. From a game theoretical point of view, such an issue, in general, is known as the *price of anarchy* [23] or the *price of stability*, depending on whether the social optimum is compared with the worst or else with the best equilibrium outcome. In simple terms, ConGaS actually selects a subset of Nash equilibria; in this case, one of the aims is to compare the average outcome of such equilibria with respect to the average over *all* equilibria. Basically, (average) equilibrium outcomes (referred to as "equilibrium") is implemented simply by ignoring restriction mechanism Rm1-2. Specifically, at any stage $t \geq 0$, each peer $i \in N$ who still misses some unit randomly selects some valid forwarder $j \in N_0$, if any, and then receives the oldest unit in $C_j^t \backslash C_i^t$. Whenever the number of receivers exceeds that of forwarders, the probability of being among those who receive (and thus also that of being among those who do not) is the same across all potential receivers. Finally, ConGas is also compared with a basic gossip protocol (referred to as "disequilibrium"). In this case, not only restriction mechanism Rm1-2 is ignored, but also and most importantly congestion can cause a peer to miss one

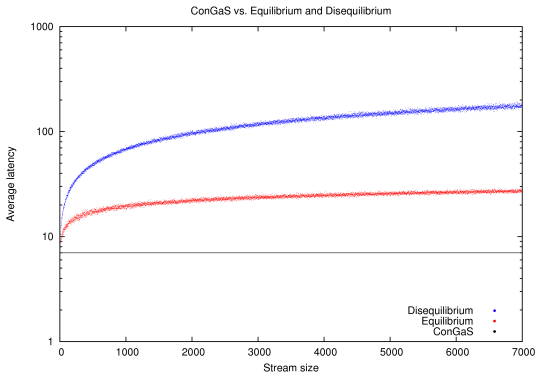

Fig. 1. Average latency: number of peers=2^8, varying stream size. y-axis log-scale

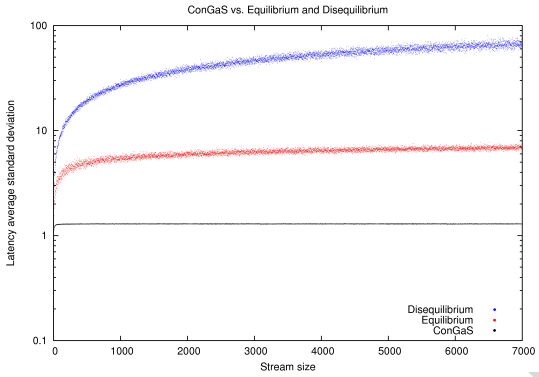

Fig. 2. Latency average standard deviation: number of peers=2^8, varying stream size. y-axis log-scale

or more slots, in which it will not receive any content unit. Our metrics, used to compare the three considered schemes are *latency* (measured in rounds) and its *jitter*. Latency Λ is obtained such that, by picking at random both a peer and a unit c^t, the former receives the latter (on average) in round $t+\Lambda$. Jitter is the variance across peers of the average latency (over all units) they experience during the whole streaming.
Figure 1 shows the behavior of the three compared streaming protocols/algorithms with a fixed number 2^8 of peers and increasing stream size. Latency with ConGaS does not depend on stream size. Moreover, such a result is optimal given the assumptions that only one unit can be sent/received by each peer in every round. Conversely, both equilibrium and (*a fortiori*) disequilibrium results deteriorate substantially as the size increases. It is worth noting that the values in the y-axis are reported in a logarithmic scale. In detail, the gap between equilibrium and disequilibrium is substantial. Figure 2 reports the (average) standard deviation of latency, which is much more limited with ConGaS rather than with the other two plots. Here again, values are reported in log-scale, and equilibrium performs much better than disequilibrium. Another significant result is that dispersion is minimal with ConGaS, while it is notable with the other two P2P exchange mechanisms. We claim this is an important result for the implementation of viable live streaming systems, as it corresponds to a jitter

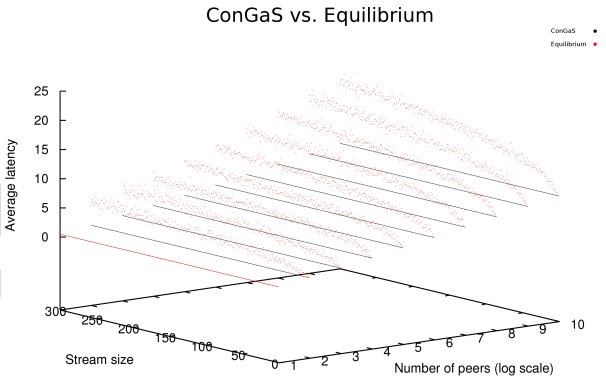

Fig. 3. Average latency: varying number of both peers and stream size.

reduction, which is a main requirement in most multimedia systems.

As disequilibrium performance is evidently very poor, in the next experiment we omit it (for better appreciating the two remaining schemes). In particular, Figure 3 shows the average latency, measured in rounds, experienced by peers in receiving any content unit, when varying both the number of peers (reported here in $\log_2$ scale) and the *size* of the stream $T+1$. As predicted by our analytical results, ConGaS obtains very stable results in terms of latency (black lines), i.e. the length of the stream has no evident impact on performance. Conversely, when considering generic (or non-restricted through Rm1-2) equilibrium outcomes (light gray points), the experienced average latency increases with number of content units. The same also happens when the number of peers growth.

VI. Implementation Issues

In this section we discuss on some issues related to the factual development of ConGaS in a real live-streaming scenario. The are at least five points to be discussed: i) how peers coordinate and cooperate, based on a shared knowledge of the content distribution process, ii) the use of a round-based time model, iii) the assumption that a single content unit can be sent/received per each round, iv) how to make the protocol able to cope with dynamic join/leave of peers in the system, v) how the protocol is resilient to free-riding.

As to the need for cooperation, optimal content distribution requires that peers agree to employ some coordination method to guarantee, for example, that nodes involved in the distribution process do not waste their time by sending content units to some other peers that do not need those contents (e.g. they already received them). Many peer-to-peer solutions adopt strategies to bypass the problem, for instance by organizing the set of peers as a tree, multi-tree, or even employing more sophisticated approaches (e.g. DHTs) [4], [7], [8]. From this standpoint, our approach is more dynamic, since peers act by changing their partners at each round, choosing them at random. This has the advantage of augmenting the fairness of the protocol. The cost is that peers must share the same seed employed to construct the pseudo-random number sequence

that regulates the content distribution. However, we claim that this does not introduce any particular complications which prevent a real implementation of the scheme.

The use of rounds is widely utilized in many distributed algorithms and communication protocols, even when streaming applications are built on top of them, e.g. [1]. Of course, in our approach we do not require perfect synchronization among peers' clocks, neither there is a need for employing strict synchronization barriers. Rather, a loose synchronization (for example based on NTP) is sufficient to guarantee that peers are not involved in multiple content units' transmissions (or receptions) at the same time.

In our model, we assume that peers can send/receive only one content unit per round. This allows to simplify the theoretical analysis and the resulting protocol. If peers in the system have similar networking resources with comparable performances, such an assumption does not introduce any limitation on the goodness of the protocol. When very heterogeneous devices are involved (using different networking technologies), such as in a mobile scenario, for instance, such an assumption could limit the the performances of the distribution process. We plan to extend in the future our model to consider the case when heterogeneous peers are present in the system.

Another important issue to consider in distributed peer-to-peer applications is the typical dynamism of peers that join and leave the system, hence imposing an adaptive configuration of the peer-to-peer architecture. In this work, we did not specifically presented schemes to cope with this problem. In this respect, we are currently investigating methods to group peers (that would represent a single player in the theoretical model). This way, ConGaS can be safely executed within each peer group, while the use of groups of peers as player in the congestion game would certainly provide higher tolerance to faults and churns.

Finally, participants are assumed to fully cooperate, in that they all satisfy one request whenever they receive some valid one. A main tool for limiting free-riding is the capability to tag those who do not cooperate, so that everybody can recognize them. In this respect, our proposed method constructs a new P2P network for each content unit to be distributed, and thereby each peer is expected to interact with all other peers several times during the whole course, therefore incentive-based schemes are applicable [24]. This should also discourage peers to maliciously exploit the shared seed employed during the pseudo-random number generation, implementing full knowledge of the game course.

VII. Conclusions

In this paper we modeled P2P real-time streaming by means of multistage congestion game forms. Under the assumptions that at each round each peer can at most send/receive a single content unit, then P2P content exchange can be scheduled so that *each unit* reaches everybody in $m + 1$ (consecutive) rounds/stages, with corresponding whole duration $T + m + 1$.

We provide a strategy restriction mechanism which, for any (per stage) content distribution over peers, restricts the set of feasible forwarders. Although simple, this mechanism is very useful: it impedes all those requests submitted by selfish peers at equilibrium which must be ignored to achieve minimal duration. This allows to fasten the stream distribution to all peers, and to minimize congestion. Based on this machinery, we provided an algorithm (ConGaS); our simulations confirm the viability and the goodness of the proposal.

References

[1] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "Bar gossip," in *Proc. of OSDI '06*, 2006, pp. 191–204.

[2] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 325–336, 2008.

[3] Z. Cai and X. Lin, "Qcast: A qos-aware peer-to-peer streaming system with dht-based multicast," in *Proc. of GPC 2008*, 2008, pp. 287–295.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, 2003.

[5] Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving VoD server efficiency with bittorrent," in *Proc. of Conf. on Multimedia*, 2007.

[6] J. Venkataraman and P. Francis, "Chunkyspread: Multi-tree unstructured peer-to-peer multicast," 2006.

[7] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. of SIGMETRICS'00*.

[8] J. Peltotalo, J. Harju, A. Jantunen, M. Saukko, and L. Väätämöinen, "Peer-to-peer streaming technology survey." in *ICN*. IEEE Computer Society, 2008, pp. 342–350.

[9] D. Pompili, C. Scoglio, and L. Lopez, "Multicast algorithms in service overlay networks," *Comput. Commun.*, vol. 31, no. 3, pp. 489–505, 2008.

[10] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *NOSSDAV '02*. ACM, 2002, pp. 177–186.

[11] X. Zhang, J. Liu, B. Li, and Y. S. P. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," 2005, pp. 2102–2111 vol. 3.

[12] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into pplive: A measurement study of a large-scale p2p iptv system," in *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[13] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "Anysee: Peer-to-peer live streaming," in *Proc. of INFOCOM*, April 2006.

[14] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE JSAC*, vol. 20, no. 8, pp. 1489–1499, 2002.

[15] C. Chekuri, J. Chuzhoy, L. Lewin-Eytan, J. S. Naor, and A. Orda, "Non-cooperative multicast and facility location games," in *Proc. of 7th ACM conference on Electronic commerce*, 2006, pp. 72–81.

[16] W. Lin, H. Zhao, and K. Liu, "A game theoretic framework for incentive-based peer-to-peer live-streaming social networks," in *Proc. of Acoustics, Speech and Signal Processing*, 2008.

[17] M. Meo and F. Milan, "A rational model for service rate allocation in peer-to-peer networks," in *Proc. of INFOCOM'05*, 2005.

[18] Z. Yang and H. Ma, "A game-based mechanism for avoiding routing hotspot in p2p streaming distribution," in *Proc. of Multimedia and Expo*, 2007.

[19] M. Y.-K. K. Yeung, "Game-theoretic scalable peer-to-peer media streaming," in *Proc. of IPDPS'08*, 2008.

[20] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, pp. 124–143, 1996.

[21] R. Holzman and N. Law-Yone, "Strong equilibrium in congestion games," *Games and Economic Behavior*, pp. 85–101, 1997.

[22] G. Rossi, S. Ferretti, and G. D'Angelo, "Equilibrium selection via strategy restriction in multi-stage congestion games for real-time streaming," University of Bologna, Tech. Rep. UBLCS-2009-11, 2009, www.cs.unibo.it/pub/TR/UBLCS/2009/2009-11.pdf.

[23] C. Chekuri, J. Chuzhoy, L. Lewin-Eytan, J. S. Naor, and A. Orda, "Non-cooperative multicast and facility location games," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 6, pp. 1193–1206, 2007.

[24] T. Ngan, D. S. Wallach, and P. Druschel, "Incentives-compatible peer-to-peer multicast," in *Proc. of 2nd Workshop on Peer-to-Peer Systems*, 2004.