A Markov model for the evaluation of cache insertion on peer-to-peer performance

Ioanna Papafili and George D. Stamoulis

***Abstract*—Peer-to-peer file sharing applications generate huge volumes of the Internet traffic, thus leading to increased congestion and costs for the ISPs, particularly due to inter-domain traffic. Thus, analysis of peer-to-peer applications and related optimization approaches (such as locality awareness or caching techniques) has been the subject of extensive recent research. In this paper we introduce and analyze a probabilistic model that employs a Markov chain, aiming to approximate the transient evolution of a swarm with a fixed number of peers. This model estimates the distribution of the number of chunks already downloaded by a certain peer at any time. We also show how this model can serve as a tool to analyze certain properties of peer-to-peer applications, such as monotonicity of performance, and primarily to evaluate the effectiveness of cache insertion in a network serving peer-to-peer. For tractability reasons, the model employs certain simplifications of the original BitTorrent protocol, the impact of which is limited as validated experimentally.**

***Index Terms*—peer-to-peer, evaluation, Markov chain, performance.**

I. Introduction

Peer-to-peer file sharing systems generate huge volumes of Internet traffic, which lead to increased costs for the ISPs, particularly for inter-domain traffic. Several approaches have been proposed that aim to achieve reduction of the inter-domain traffic and/or the download times. Most of them try to increase the degree of localization of traffic, thus reducing inter-domain traffic and possibly improving performance. Such approaches either propose biased peer selection [1], or using an Oracle service for peer ranking [2] or using existing information from content distribution networks [3]. In [4], Papafili *et al.* introduced a different approach, namely the insertion of resourceful overlay entities (referred to as ISP-owned Peers, IoPs). IoPs participate actively in the peer-to-peer swarm and exchange data chunks with regular peers, thus serving as transparent and non-intercepting caches, which however initially have to acquire the content too.

In this paper, we focus on BitTorrent [5] as it is the most popular peer-to-peer file sharing application; however, our study can be also extended to bittorrent-like streaming applications such as Tribler. In particular, we develop a Markov model for a BitTorrent swarm with a fixed number of peers. This model estimates approximately the transient distribution of the number of chunks already downloaded by a certain peer at any time.The model can serve as a tool to analyse certain properties of BitTorrent and to evaluate the impact on performance of cache insertion in the form of an IoP. It should also be noted that for the latter purpose a model for the transient evolution of the swarm is much more suitable, because the insertion of an IoP mainly aims to have a positive impact in the beginning, where most of the leechers in a swarm are lacking the largest part of the shared content. Once this purpose is attained, then the IoP can devote its resources to another swarm, rather than remain in the previous one.

The remainder of this paper is organized as follows: In Section II, a brief overview of articles on BitTorrent modeling is presented. In Section III, we propose a Markov model for the evaluation of cache insertion in BitTorrent networks; motivation, assumptions and evolution are described here. In Section IV, results for simulative evaluation of the approximating assumptions of the model are presented; then, we validate the Markov model's results with results derived by means of simulations; and finally, we present evaluation results acquired by use of the Markov model on the impact of cache insertion in BitTorrent network. Last, in Section V, we conclude our contribution. Detailed equations of the Markov model are given in the Appendix.

II. Related Work

There is an extensive literature on peer-to-peer performance evaluation. Many relevant works are exclusively based on analysis of measurements from actual systems and/or simulations. In this subsection, we overview selected articles that include models for the performance evaluation of peer-to-peer file-sharing with emphasis on BitTorrent and we comment on their relation with our work. . Evaluation results attained by models that apply to the steady-state of a BitTorrent swarm cannot be compared to results attained by our model. On the other hand, comparison of our results with those attained by models for transient analysis is meaningful and is presented at the end of Section IV.

In [6], Kumar and Ross analyze the minimum distribution time for a file in a system with seeds and leechers. In particular, by employing a deterministic fluid-flow model, they provide a lower bound that involves the download and upload rates of the various peers and then show that this bound can indeed be achieved by scheduling the various transfers of the file appropriately.

Manuscript received March 26, 2010. This work was supported partially by the EU IST project SmoothIT (FP7-ICT-216259).

Ioanna Papafili is a PhD student in Athens University of Economics and Business, Patission str. 76, 10343, Athens, Greece (phone: 00306970015650; e-mail: iopapafi@aueb.gr).

George D. Stamoulis is a professor in Athens University of Economics and Business (e-mail: gstamoul@aueb.gr).

In [7], Yang and de Veciana initially deal with the capacity attained in peer-to-peer systems due to their fundamental feature that each peer can serve other peers while still downloading the missing content. The authors thus develop a simple deterministic model that shows the effect of this feature in the transient case similar to that of our model, with only one of the peers storing initially the content file. In particular, it follows that the average delay per peer is logarithmic in the number *N* of peers. Moreover, if the file is partitioned into *m* chunks, then due to pipelining, the average delay is reduced by a factor of *m*. The authors of [7] also develop a two-dimensional Markov model for the steady state analysis of the system. The problem of calculating the completion time is also studied by Mundiger *et al*. in [8], under more general assumptions. These authors derive the optimal centralized upload schedules both for the case of a central server and for the case of a decentralized system with all peers having equal capacities. They then develop another model for the transient evolution of a peer-to-peer system, whereby the number *N(t)* of peers that have already downloaded the file by time *t* is modeled as an age-dependent branching process with a family size of 2 in each generation. Therefore, the expected value *E[N(t)]* grows exponentially with time *t*, provided that there is sufficient demand in the system.

In [9], Qiu and Srikant initially present a deterministic fluid model for the performance of BitTorrent. The model of [9] is motivated by the Markov model of [8] and comprises the same parameters. The authors present (among others) a probabilistic model for the evaluation of the parameter η that expressed the effectiveness of BitTorrent in the sense of the degree of the contribution of each downloader to the other ones. In particular, this model quantifies the probability η that a particular downloader has a chunk that is among the ones needed by another one. In fact, the assumptions made with respect to the distribution and selection of chunks possessed by a peer are the same to those our model, as described in Section III.B. It turns out that $\eta \approx 1-\left(\log N/N\right)^k$, where *k* is the total number of chunks of the file. This implies that for a large file (i.e. for a large value of *k*), $\eta \approx 1$; that is, a downloader contributes to the others almost as much as a seed.

In [10], Leibnitz *et al*. present a fluid flow model for evaluating the transient performance, in particular reliability and efficiency, of content distribution services that can be realized by traditional client/server (C/S) architectures or peer-to-peer networks involving malicious peers.

Besides [8], several articles deal with the steady-state performance analysis of BitTorrent with dynamically varying population. Next, we briefly overview selected such articles. In [11], Ge *et al.* present a steady-state queueing model that comprises all the main ingredients of a peer-to-peer file sharing system, while applies to a variety of such systems. This model is then solved analytically by means of an approximation based on bottleneck analysis, and it is validated by means of simulations. The work of Fan *et al.* in [12] deals with an important tradeoff arising in BitTorrent, namely: achieving fast downloads vs. keeping "fat" (i.e. resourceful) peers in the system as much as possible in order to help other peers attain a fast download. The latter objective appears to be unfair for the fat peers, thus giving rise to a tradeoff between performance and fairness (i.e. better service for peers contributing more to the system), which is investigated in the paper by means of steady-state analysis.

III. The Markov Model

A. Motivation

As already stated, the Markov model proposed in this paper models a BitTorrent swarm, namely an overlay network for the sharing of one file. We focus on BitTorrent since it is the most popular file sharing application. The purpose of the Markov model is the analysis of certain properties of BitTorrent, such as monotonicity and scalability of performance, and primarily, the evaluation of cache insertion as a means to achieve performance improvement. As already explained in Section I, by caches we mean overlay entities that participate actively in the peer-to-peer network and exchange data chunks with regular peers. These entities are referred to as ISP-owned Peers (IoPs) [4].

A Markov chain is employed to estimate approximately the transient distribution of the number of chunks downloaded by each peer. Based on this, we can also estimate other performance measures such as the upper tail of the distribution of the time required for a peer to complete downloading a file. For the purpose of analytical tractability, the model employs certain simplifications of BitTorrent; thus, it is expected that its outcomes will constitute bounds for the corresponding metrics of the actual BitTorrent.

The Markov model in its present state can be employed for studying the following issues:

- Monotonicity of completion times in a BitTorrent network for different numbers of regular peers;
- Impact of the insertion of an IoP on performance; tradeoff w.r.t. the associated capacity of the IoP;
- Impact of insertion of mere IoPs with heterogeneous capacities; tradeoff w.r.t. the number and capacity of the IoPs.

For each one of the cases involving IoP(s), the transition probability matrix of the Markov chain is slightly different than that of the first case without IoP. Moreover, the model can easily be extended to other cases and scenarios where other optimization approaches, e.g. locality awareness, are employed.

In the subsections III-B and III-C, we present the basic assumptions, the rationale and basic equations that describe the evolution of the Markov chain.

B. Assumptions

Our Markov model is a discrete time model; i.e. time is slotted. Originally, we consider $N+1$ peers in the swarm; namely, *N* downloaders with initially 0 chunks, and 1 seeder which has all *K* chunks. Moreover, for simplicity we assume that after a downloader finishes its downloading then it turns into a seed, namely it does not leave the swarm. (By introducing minor modifications to the formulation, the model

can accommodate other assumptions on this issue.) The population of peers remains constant.

Next, we present our assumptions on the modeling of the peer-to-peer application. First, we assume random chunk selection, instead of BitTorrent's rarest first replication. Due to symmetry among chunks in their initial distribution, the system's state is fully specified by the number of chunks that each peer has acquired until the end of step *n*, or equivalently by the number of peers out of *N* that have 0, 1, …, or *K* chunks at step *n*. Therefore, the number of different states with this formulation would be equal to the number of choosing *k* elements out of *n* with repetition:

$$\binom{N+K}{K} = \frac{(N+K)!}{K!N!}.$$

Due to the prohibitively large state space we resort to an approximation, which is motivated by the fact that due to symmetry the evolution of the marginal distribution of the state any given peer is same as that of any other peer. Let D be a tagged peer out of the set of the *N* downloaders. The state of D (as well as that of any other peer) belongs to $\{0,1,2,\ldots,K\}$. The objective of the model is to derive this marginal distribution of the state of the tagged peer D. Furthermore, for simplicity reasons, we assume that at every step *n*, only 0, 1 or 2 chunks can be downloaded by each peer, regardless of how many peers have unchoked this peer; additionally, each peer can be unchoked by every other peer only once. As already mentioned chunks are selected at random and uniformly. Thus, all chunks that D is missing are considered useful and assumed to be sought simultaneously. We have assumed that a peer can download up to 2 chunks at each step since we have measured by means of simulations that the number of chunks downloaded per peer in each choking interval is in general less than or equal to 2. This is supported by the calculations above and by the experimental results presented in Section IV.

Second, BitTorrent's choking algorithm, the well-known 'tit-for-tat', is ignored; unchokes are given to peers randomly selected out of the set of potentially interested downloaders. Each peer has the possibility for *cl* unchokes; parameter *c* represents the minimum of a) the actual number of unchokes that each peer possesses (by the BitTorrent protocol) and b) of the number of unchokes possible due to the upload bandwidth capacity bottleneck in this peer. For instance, the protocol allows each peer to unchoke up to $cl = 5$ other peers per choking interval. In order to derive results comparable with results attained by the simulations, we take $cl = 2$. For instance, a peer with nominal upload bandwidth equal to 512 kbps is able to upload/serve only 2 chunks. Indeed, 512 kbps corresponds to 64 KB/s upload. In an interval of 10 secs, this corresponds to uploading 640 KB, which is slightly higher that 2x256 KB (where 256 KB is the size of a chunk).

Due to the aforementioned assumptions, our Markov model corresponds to a version of BitTorrent where all decisions are made randomly, and thus is expected to have inferior performance compared to the original BitTorrent. Consequently, the estimates obtained by our model are expected to constitute upper bounds of the actual performance of the BitTorrent protocol.

C.Evolution of the Markov Chain

First, we consider modelling the native BitTorrent protocol. The transient marginal distribution of the state of a regular peer D at step *n* is denoted $P(n) = [P_n(0), P_n(1), \ldots, P_n(K)]$, where $P_n(k) = \Pr[X_n = k]$. The transient distribution at step *n*+1 is derived as follows:

$$P_{n+1}(k) = P_n(k-2)P_{n+1}(k-2,k) + \\ + P_n(k-1)P_{n+1}(k-1,k) + P_n(k)P_{n+1}(k,k)$$

where the transition probability $P_{n+1}(k-2,k)$ corresponds to the event that peer D is unchoked by and finds useful chunks in two or more other peers at step *n*+1, given that peer D has $k-2$ chunks at the end of step *n*; transition probability $P_{n+1}(k-1,k)$ is defined similarly, while $P_{n+1}(k,k)$ is the transition probability that peer D is either choked by all peers or does not find any useful chunk at any peer he is unchoked by, given that it has *k* chunks at step *n*. The three transition probabilities sum to 1.

The calculation of the transient probabilities at every step is an iterative process; the procedure is depicted in Fig. 1. Due to the fact that population size *N* is fixed, there is a deterministic upper bound in our setting for the overall completion time. Since the completion time of the last peers might be quite large, this upper bound can be loose. Thus, we choose to consider the *G*-th percentile of the overall completion time; that is step *n**, when a large portion of the peers (say 90% or 95%) will have finished downloading; that is, they have *K* chunks. Using the transient marginal distribution of the state of D, it follows for $n^*: n^* = \min\{n : P_n(K) > G\}$, where *G* will be taken either 0.90 or 0.95. Additionally, we also consider the average completion time among all peers, which is $\bar{n} := \sum_{n=1}^{\infty}(1 - P_n(K))$.

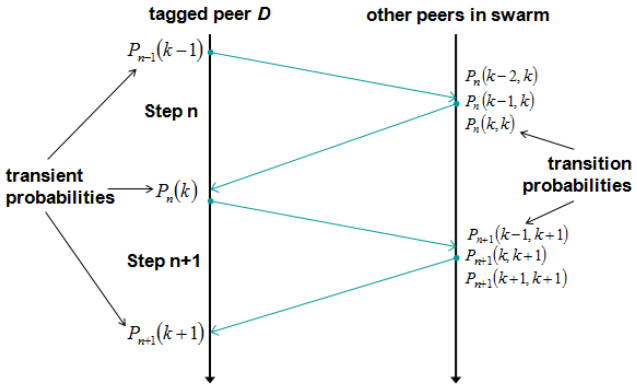

Fig. 1. Iterative process.

The equations that characterize the Markov chain evolution are given in the Appendix. These equations involve also the distribution of: a) the number $N_s(n)$ of downloaders that have *K* chunks at step n, and b) number $N_e(n)$ of downloaders that have 0 chunks at step *n*. Note that the corresponding sets of downloaders are non-overlapping. Since *K* is large, initially $N_s(n) \approx 0$, while as time progresses $N_e(n) \approx 0$. As an approximation, these random variables are taken to be independent. Moreover, their distributions are approximately taken as binomial, i.e.:

$$P(N_s(n)=x)=\binom{N+1}{x}(P_n(K))^x(1-P_n(K))^{N-1-x},$$

$$P(N_e(n)=z)=\binom{N+1}{z}(P_n(0))^z(1-P_n(0))^{N-1-z}.$$

This amounts to taking that the event where a certain peer is in state 0 (or resp. in state K) is independent of the corresponding events for other peers; note that all these events are identically distributed.

Next, we consider the insertion of an IoP with higher capacity than regular peers in the BitTorrent overlay. For this case, the transition probability matrix of the basic Markov chain is modified as follows: Due to the higher capacity, we assume that the IoP is always unchoked twice by the original seed; namely the IoP downloads 2 chunks at every step with probability 1, both of them from the original seeder. Note that the IoP behaves as a regular peer until step $n = K/2$, and from that step ahead as a seeder. In order to further study the impact of the IoP on peer-to-peer performance, we also considered a third case, namely, the insertion of the ISP-owned Seed (IoS) scenario, i.e. the insertion of a second seeder again with higher capacity than the original regular one. Performance improvements achieved by the insertion of the IoS are expected to constitute an upper bound to the performance improvements achieved by the insertion of IoP.

IV. Experimental results

In this section, we present experimental and numerical results: a) validating the approximating assumptions of the model, b) assessing the accuracy of the model, c) evaluating peer-to-peer performance with and without IoP

A. Validation of approximating assumptions

First, we present experimental results in order to validate the approximating assumptions of the model (see Subsection III.B). These assumptions are validated by means of simulations of a BitTorrent implementation [13] in the *ns-2* simulator [14]. Simulation setup includes a topology with 2 equally sized ASes, interconnected by means of an inter-domain link. Each peer has an access link of 4096/512 kbps download/upload bandwidth, while the content file size is 80 MB. Moreover, we have taken that all peers, including the original seed, start at the same time instant (0.0 secs). In our study, small and medium swarm sizes have been considered e.g. with $N = 20,30,\ldots,160$ peers. Indeed, in reality, a significant portion of swarms has such populations, according to the measurements of the sizes of actual swarms in the Internet presented in [15].

Two-step transition Markov chain. The Markov model restricts the maximum possible number of downloaded chunks by a peer per step to 2. The reason for this assumption is two-fold: 1) It does not give rise to too complicated equations, while 2) it is a much more realistic and accurate assumption than taking the maximum number of downloaded chunks by a peer per step to equal 1.

In order to decide on adopting this assumption, we monitored the number of the new chunks that every peer obtains during each choking interval in the regular BitTorrent where this restriction does not apply. We focus on an experiment with a swarm having 50 peers monitored for a period of 50 slots. Approximately 70% of the 2500 samples of the number of new chunks downloaded per peer and per slot were equal to 0, 1 or 2. Fig. 2 depicts the average number of chunks downloaded by all 50 peers of the swarm in each choking interval. Note that only in less than 20% of the cases the average number of chunks is greater than 2, yet it is still less than 2.5 in all such cases. The average number of downloaded chunks over all peers equals 1.449 < 2. Thus, the underlying assumption of maximum 2 chunks downloaded per choking step is satisfactorily accurate.

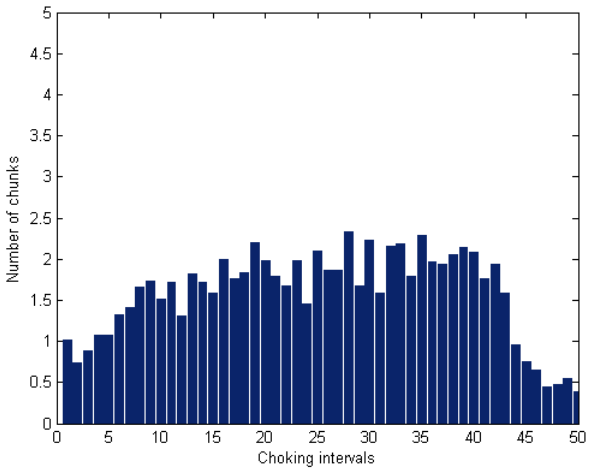

Fig. 2. Average number of chunks downloaded per choking interval over all peers.

Native vs. randomized BitTorrent. Performance achieved in native BitTorrent is compared here to performance achieved in randomized BitTorrent. By native BitTorrent it is meant that the choking algorithm and rarest first replication are employed as peer and chunk selection methods respectively; while by randomized BitTorrent it is meant that both peer and chunk selection processes are performed at random. Note that randomized implementation of BitTorrent is approximated by the assumptions of the Markov model.

We have run simulations for the implementation of the native BitTorrent protocol and the randomized one and have compared them in terms of completion times. In Fig. 3, the average completion times for swarms of $N = \{20,30,\ldots,100\}$ peers, are presented for the two implementations. Note that regardless of the swarm size increase, the difference in completion times between the two implementations remains almost fixed and bellow 11%; this also holds when other approaches are employed such as locality awareness, or cache insertion. Therefore, performance difference between randomized and native BitTorrent remains fixed. Comparison of results derived by simulations and by the Markov model is presented and discussed in Section IV.B.

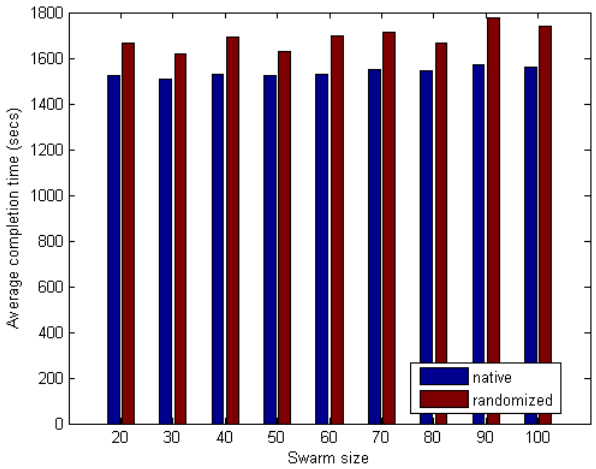

Fig. 3. Average download completion times for native BitTorrent (blue), and randomized BitTorrent (red).

B. Verification of the model

Comparison of simulation results vs. results derived by the Markov model. We consider the same simulation setup as in Section IV.A To compare the completion times derived by simulations (continuous time) to results (steps) derived by calculation of the Markov chain in Matlab [16] (discrete time), we have transformed discrete results to continuous, namely by converting time measured in steps to seconds. As mentioned above, each step of the Markov model corresponds to one choking interval, which has duration 10 secs. [5]. Therefore, to transform steps to seconds, we just need to multiply number of steps by the duration of the choking interval, e.g. 10 secs. Fig. 4 depicts simulation results for native BitTorrent (blue), simulation results for randomized BitTorrent (green) and results derived by the Markov model for $cl = 2$ multiplied by 10 secs (red). We observe that relative difference between calculated results and simulation results (for the *native* bittorrent implementation) lies under 5% (except for the case of $N = 20$ swarm); this also holds for different setups, although the relevant results are not presented due to space limitations. Thus, the Markov model approximates satisfactorily the native version of BitTorrent.

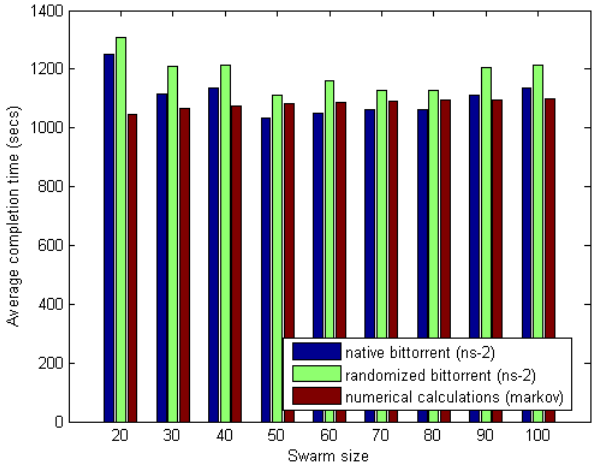

Fig. 4. Comparison of average completion times derived by simulations vs. average completion times (steps×10 sec.) derived by the Markov model.

C. Evaluation results for peer-to-peer properties

Next, we present numerical results calculated according to the Markov model in Matlab. As stated in Section III.A, one of objectives of the Markov model is to study properties of the BitTorrent protocol such as monotonicity w.r.t. the swarm size and impact of the original seeder's capacity.

Monotonicity. We considered swarms with $N = \{10,15,20,25,\ldots,160\}$ peers, each having capacity $cl = 2$, and one original seeder with upload capacity $cs = 2$. The file size is taken 40 MB, thus consisting of approximately 160 chunks of 256 KB each. Note that the Markov chain for this file size has 161 possible states, e.g. $\{0,1,2,\ldots,K\}$, and that the first step when $P_n(K)$ is non-zero is $n = 81$. This lower bound follows from the fact that up to 2 chunks per peer can be downloaded at each step, but only 1 chunk can be downloaded from the unique original seeder in the 1st step, while the total number of chunks is 160. We calculated the *G*-th percentile of the completion time *n** for $G = \{0.90,0.95,0.99\}$. Fig. 5 depicts *n** for different *N*'s. Observe that *n** increases only slightly (almost remains stable) as the swarm size increases, which verifies scalability of this peer-to-peer protocol.

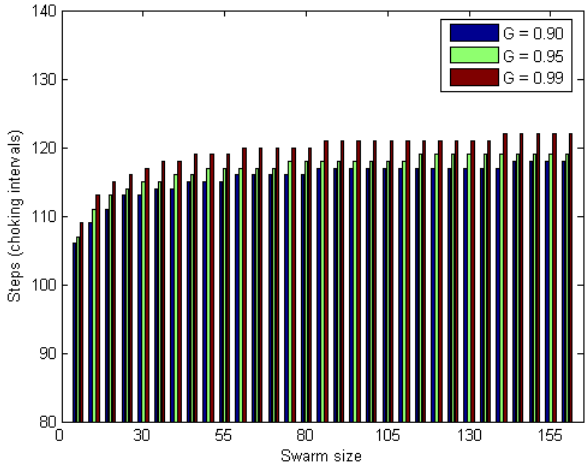

Fig. 5. *G*-th percentile *n** for *N*={10,15,20,…160} when *G*={0.90,0.95,0.99} derived by the Markov model in Matlab.

Impact of the original seeder's capacity. In [1], Bindal *et al.* argue that the original seeder's capacity has important impact on the download times. In particular, they observe that their biased neighbour selection may deteriorate the completions times. However, they show that the greater the seeder's capacity, the less the impact of biased neighbour selection on the completion times. Furthermore, in [17] Le Blond *et al.* argue that the original seeder's capacity is critical to the high chunk diversity (higher probability for the peers to find useful chunks), which impacts the overhead (amount of content that crosses an inter-ISP link) and peers' slowdown (the experimental peer download completion time normalized by the ideal completion time).

We considered a swarm with $N = 100$ peers with upload capacity $cl = 2$, and one original seeder with varying upload capacity $cl = c$. The file size is again taken to be 40 MB, and thus the model has 161 again states. Fig. 6 presents the G-th

percentile n^* for $G=\{0.90,0.95\}$ attained for different values of seeder's capacity $c=\{2,4,\ldots,20\}$. Note that as the seeder's capacity increases, n^* decreases with constant rate, which is in agreement with the aforementioned results of [1] and [17].

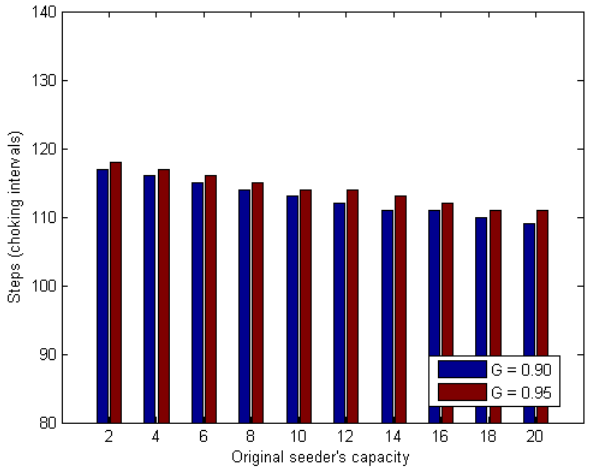

Fig. 6. G-th percentile n^* for c={2,4,6,…20} when G={0.90,0.95} derived by the Markov model in Matlab.

D.Evaluation of IoP cache insertion in Bittorrent network

Next, we present numerical results calculated according to the Markov model in Matlab for the evaluation of cache insertion. As stated in Section III.A, the main objective of the Markov model is to serve as an evaluation tool for certain optimization approaches of peer-to-peer networks, such as cache insertion. Below, we present and discuss the impact of cache insertion on the overall completion times of the whole swarm and the on upper tail distribution of an individual peer.

Evaluation of IoP cache insertion. In [4], Papafili *et al.* have shown by means of simulations on the *ns*-2 that the insertion of the IoP in one of the ASes participating in a swarm results in important reduction of the completion times of the peers in this AS. Furthermore, it was shown that significant inbound inter-domain traffic reduction is also achieved for this AS. However, in its current version, our Markov model does not incorporate any topology information, and therefore cannot lead to conclusions on inter-domain traffic. Extending the model appropriately so as to provide estimates for the volume of inter-domain traffic is a direction for our future work. Thus, our main objective is to show that the proposed Markov model validates simulation results that reveal the positive impact of the insertion of the IoP on the download times. We considered the same setup as in Section IV.A for three scenarios: a) no cache insertion, b) insertion of the IoP, and c) insertion of the IoS. IoP/IoS are assumed to have upload capacity $cp=10$.

Fig. 7 depicts the 95-th percentile ($G=0.95$) of the overall completion time for the three aforementioned scenarios. Observe that the insertion of IoP improves significantly the overall completion time especially for small or medium to small swarms, whereas the insertion of IoS achieves even higher reduction. As expected, the performance achieved by the insertion of IoS constitutes lower bound for the performance achieved by the insertion of IoP. It should be noted that a rather moderate value for the IoP/IoS capacity has been chosen. This also has been kept fixed as the swarm size increases, hence the deterioration of the improvement attained. Note that in [4], the upload capacity of IoP was 20 times greater than that of regular peers, whereas here it is only 5 times greater. An ISP could achieve even better performance improvements for larger swarms by either increasing the capacity of the IoPs/IoS or introducing more such 'small' IoPs/IoSs in its domain. Notice that despite its advantage the IoS insertion is only slightly better than the IoP, since the phase when the cache acts as leecher is too short compared to the swarm's lifetime. Similar results apply also for $G=0.90$.

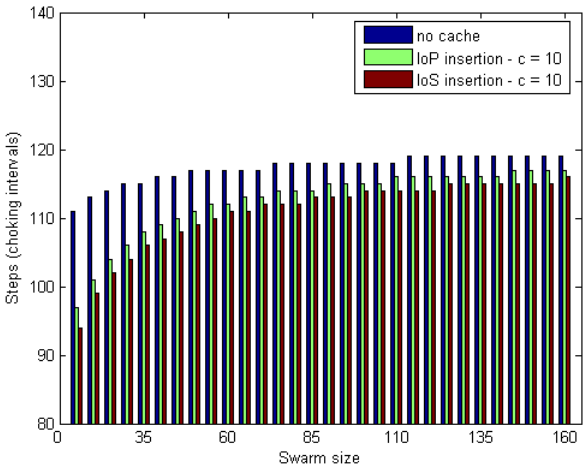

Fig. 7. Comparison of 95-th percentile of completion time for a) no cache insertion, b) insertion of IoP, and c) insertion of IoS.

Fig. 8 depicts the G-th percentiles of the overall completion time of the swarm for different values of upload capacity of the IoP. We observe that the percentile is improved significantly, up to 13%, when the capacity of the IoP is $cp=20$, namely 10 times the capacity of the regular peers. Since this is a moderate value for the upload capacity of the IoP, we expect even lower completion times for higher values of upload capacity.

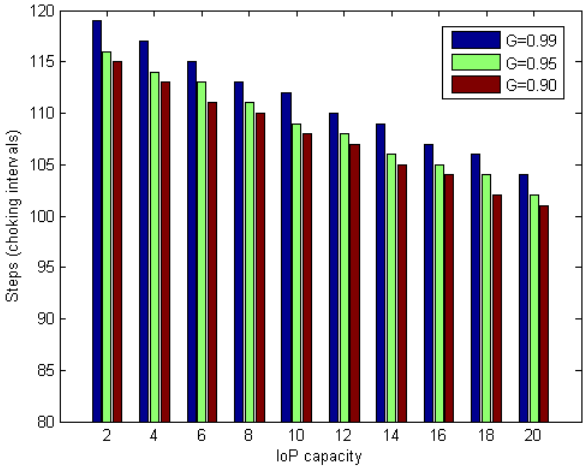

Fig. 8. Comparison of G-th percentiles of completion time for upload capacity of the IoP cp={2,4,6,…,20} for G={0.90,0.95,0.99}.

Finally, we compare the estimates of completion times derived in [6] and [7] with our outcomes. Indeed, [7] is based on the assumptions of unconstrained download capacity and optimal scheduling of chunk selections and uploads; so that uploading of chunks from one leecher to the others is exploited

as much as possible. These would give in our case a completion time per peer of $(\log_2 N + (2K-1)/2)/cs$, which is a very loose lower bound; e.g. for $N=100$, $K=161$, and $cs=2$, this equals 83.41 which is almost equal to the lower bound of 81 steps calculated by the Markov model. Also, [6] is based on optimal scheduling; for the cases of Fig. 4, [6] would give the straightforward lower bound of 80, since each peer can download at most 2 chunks per slot and the total number of chunks is 160.

V. Conclusion

We have developed and studied a probabilistic model that employs a Markov chain to approximate the transient evolution of the BitTorrent peer-to-peer file sharing network,. This model aims at studying performance properties of the BitTorrent protocol, as well as at evaluating optimization approaches applicable to peer-to-peer networks such as cache insertion. The model estimates the transient distribution of the number of chunks downloaded by each given peer and from this other performance measures such as the upper tail of the distribution of the time required for an individual peer to complete downloading a file can be also computed.

For tractability, we have adopted certain simplifications of BitTorrent, such as randomized peer and chunk selections. Additionally, we have implemented in *ns*-2 the simplified version of BitTorrent and have compared the performance achieved thereby to that of the native BitTorrent. Furthermore, we have compared native implementation simulation results to numerical results calculated by the Markov model and have verified that the Markov model approximates it satisfactorily.

Using our model, we have derived numerical results that reveal monotonicity of performance of the BitTorrent protocol with respect to the swarm size. Moreover, we have observed that results from recent research on the impact of the original seeder's capacity on completion times are also verified by the Markov model. Finally, we have shown that the insertion of an ISP-owned Peer or Seed (even with moderate value of cache upload capacity) considerably improves download completion times. This is in agreement also with simulation results presented in [4], as well as with estimation of other theoretical models as those presented in [6] and [7].

Therefore, the Markov model has proven to serve as a useful design tool for the evaluation of certain optimization approaches of the performance of certain peer-to-peer protocols. Compared to other models in literature that perform transient analysis of peer-to-peer systems, our model is more suitable for evaluation of approaches that consider increase or decrease of the capacity of the system, while it can also be easily extended to consider exchange of information; future work in this direction can include extension of the model to assess the impact of locality awareness.

Appendix – Evolution of the Markov chain

Step 0: D has exactly 0 chunks: $P(0)=[1,0,\ldots 0]$.

Step 1: D can be unchoked only by the seed:

$P_1(0)=P_0(0)P_1(0,0)=P_0(0)(1-CS/N)$,

$P_1(1)=P_0(0)P_1(0,1)=P_0(0)CS/N$,

$P_1(2)=P_1(3)=\ldots=P_1(K)=0$.

Step 2: D can be unchoked either by the seed or the peers that were unchoked in step 1:

$P_2(0)=P_1(0)P_2(0,0)=P_1(0)(1-CS/N)(1-CL/N)^{CS}$,

$P_2(1)=P_1(0)P_2(0,1)+P_1(1)P_2(1,1)$, where

$P_2(0,1)=CS/N(1-CL/N)^{CS}+$

$+(1-CS/N)CS\,CL/(N-1)(1-CL/N)^{(CS-1)}$,

$P_2(1,1)=(1-CS/N)(1-CL/N\cdot Q_2(1))^{(CS-1)}$.

Respectively, transition probabilities are calculated for $P_2(2)$ and $P_2(3)$ using probabilities $Q_2(2)$ and $Q_2(3)$. Note that the terms $Q_2(1)$, $Q_2(2)$ and $Q_2(3)$ are special cases of $Q_{n+1}(k)$ which is the probability for a peer to find a useful chunk given that it is unchoked by another peer and it has *k* chunks at the beginning of step *n*+1. Term $Q_{n+1}(k)$ is derived at the end of the Appendix.

Step n: Let $P(n)=[P_n(0),P_n(1),\ldots,P_n(K)]$ be the marginal distribution of the state of D at step *n*.

Step n+1: The number $N_s(n)$ of downloaders influences the contention among the remaining downloaders; thus it should be taken into account. We distinguish two cases here: a) $n\le K$: $N_s(n)=0$; there are *N* downloaders and only one seeder in the swarm, and b) $n>K: N_s(n)\ge 0$; some downloaders may have finished downloading and are serving as seeders too. We also make use of the distribution of the number $N_e(n)$ of peers that have no chunks, since they cannot serve as sources of chunks for D.

For $k=0$: $P_{n+1}(0)=P_n(0)P_{n+1}(0,0)$, where

$$P_{n+1}(0,0)=E_{N_e(n),N_s(n)}\left[\left(1-\frac{CS}{(N-N_s(n))}\right)\left(1-\frac{CL}{(N-1-N_s(n))}\right)^{N-1-N_e(n)}\right].$$

When $n\le K$, then:

$$P_{n+1}(0,0)=\left(1-\frac{CS}{N}\right)\left(P_n(0)+(1-P_n(0))\left(1-\frac{CL}{(N-1)}\right)\right)^{N-1}$$, else:

$$P_{n+1}(0,0)=\sum_{x=0}^{N-1}\binom{N-1}{x}P_n(K)^x+(1-P_n(K))^{N-1-x}\left(1-\frac{CS}{N-x}\right)\left(1-\frac{CL}{N-x}\right)^x\left(\frac{P_n(0)}{(1-P_n(K))}+\left(1-\frac{P_n(0)}{(1-P_n(K))}\right)\left(1-\frac{CL}{(N-1-x)}\right)\right)^{N-1-x}.$$

For $k=1,2,..K-1$, the transient distribution is characterized by the following equation (especially for $k=1$ the 1st term of the sum is zero):

$$P_{n+1}(k) = P_n(k-2)P_{n+1}(k-2,k) + P_n(k-1)P_{n+1}(k-1,k) + + P_n(k)P_{n+1}(k,k),$$ where

$$P_{n+1}(k,k) = E_{N_e(n),N_s(n)}\left[\left(1-\frac{CS}{(N-N_s(n))}\right)\left(1-\frac{CL}{(N-1-N_s(n))}Q_{n+1}(k)\right)\right]^{N-1-N_e(n)}.$$

When $n \le K$, then:

$$P_{n+1}(k,k) = \left(1-\frac{CS}{N}\right)\left(P_n(0)+(1-P_n(0))\left(1-\frac{CL}{N-1}Q_{n+1}(k)\right)\right)^{N-1},$$ else:

$$P_{n+1}(k,k) = \sum_{x=0}^{N-1}\binom{N-1}{x}P_n(K)^x + (1-P_n(K))^{N-1-x}\left(1-\frac{CS}{N-x}\right)\left(1-\frac{CL}{N-x}\right)^x\left(\frac{P_n(0)}{(1-P_n(K))}+\left(1-\frac{P_n(0)}{(1-P_n(K))}\right)\left(1-\frac{CL}{(N-1-x)}Q_{n+1}(k)\right)\right)^{N-1-x}.$$

Probability $P_{n+1}(k-1,k)$ is derived accordingly, however we do not present here due to space limitations. Since probabilities $P_{n+1}(k-2,k-1)$ and $P_{n+1}(k-2,k-2)$ have been calculated already for smaller states of the current step, probability $P_{n+1}(k-2,k)$ can also be easily calculated as:

$$P_{n+1}(k-2,k) = 1 - P_{n+1}(k-2,k-1) - P_{n+1}(k-2,k-2).$$

Finally, note that the term $Q_{n+1}(k)$ is the probability for tagged peer D to find a useful chunk to download from another peer, given that D is unchoked by that other peer and D has *k* chunks. Analytically, this equals to:

$$Q_{n+1}(k) = P_n(1)\left(1-\frac{1}{K}\right) + P_n(2)\left(1-\frac{1}{K}\frac{2}{K-1}\right) + \ldots + + P_n(k)\left(1-\frac{(K-k)!k!}{K!(k-1)!}\right) + \sum_{l=k+1}^{K}P_n(l) = = \sum_{m=1}^{k}P_n(m)\left(1-\frac{(K-m)!m!}{K!(m-1)!}\right) + \sum_{l=k+1}^{K}P_n(l).$$

Particularly, term $P_n(m)\left(1-\frac{(K-m)!m!}{K!(m-1)!}\right)$ expresses the probability for tagged peer D to find a useful chunk to download from another peer say D' that has m chunks, for a certain $m \le k$. This equals the probability of peer D' being in the state *m* multiplied by the probability that D' has a chunk that is different from the *k* chunks of the tagged peer D. This expression is also used in [9], and is a consequence of the assumption of random and uniform chunk selection. In the displayed equation above, the last term implies that if another tagged peer D' has even one more chunks than D, then D will find *definitely* a useful chunk to download from D' (with probability 1).

ACKNOWLEDGMENT

The authors would like to thank all SmoothIT partners for useful discussions on the subject of the paper.

REFERENCES

[1] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, A. Zhang, Improving Traffic Locality in BitTorrent via Biased Neighbor Selection, 26th IEEE International Conference on Distributed Computing Systems, p. 66, 2006

[2] V. Aggarwal, A. Feldmann, C. Scheideler, Can ISPs and P2P users cooperate for improved performance?, ACM SIGCOMM Computer Communication Review, vol. 37, pp. 29-40, July 2007

[3] D.R.Choffnes, F.E. Bustamante, Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems, ACM SIGCOMM Computer Communication Review, vol. 38, pp. 363374, 2008

[4] I. Papafili, S. Soursos, G. D. Stamoulis, Improvement of BitTorrent Performance and Inter-Domain Traffic by Inserting ISP-owned Peers, 6th International Workshop on Internet Charging and QoS Technologies (ICQT'09), Aachen, Germany, May 2009

[5] B. Cohen, Incentives build robustness in BitTorrent, In Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.

[6] R. Kumar, K.W. Ross, "Peer-Assisted File Distribution: The Minimum Distribution Time", Hot Topics in Web Systems and Technologies, 2006. HOTWEB '06. 1st IEEE Workshop on Hot Topics in Web Systems and Technologies, Nov. 2006

[7] X. Yang, G. de Veciana, "Performance of Peer-to-Peer Networks: Service Capacity and Role of Resource Sharing Policies", Performance Evaluation, 2006

[8] J. Mundinger, R. Weber, G. Weiss, "Analysis of peer-to-peer file dissemination", SIGMETRICS Perform. Eval. Rev. 34, 3 (Dec. 2006)

[9] D. Qiu, R. Srikant, "Modeling and Performance Analysis of BitTorrent-like peer-to-peer networks", Proc. ACM SIGCOMM Conference on Applications, 2004

[10] K. Leibnitz, T. Hossfeld, N. Wakamiya, M. Murata, "Peer-to-Peer vs. Client/Server: Reliability and Efficiency of a Content Distribution Service", Proceedings of the 20th International Teletraffic Congress (ITC20), Ottawa, Canada, June 2007

[11] Z. Ge, D.R. Figueiredo, J. Sharad, J. Kurose, D. Towsley, "Modeling peer-peer file sharing systems", INFOCOM 2003, 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, vol.3, no., pp. 2188-2198 vol.3, 30 March-3 April 2003

[12] B. Fan, D. Chiu, J. Lui, "The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design", Proceedings of the 2006 IEEE international Conference on Network Protocols. ICNP. IEEE Computer Society, Washington, DC

[13] Eger K., Simulation of BitTorrent Peer-to-Peer (P2P) Networks in ns-2: https://sites.google.com/site/koljaeger/bittorrent-simulation-in-ns-2

[14] The Network Simulator – *ns*-2 simulator: http://www.isi.edu/nsnam/ns/

[15] T. Hossfeld, D. Hock, S. Oechsner, F. Lehrieder, Z. Despotovic, W. Kellerer, M. Michel, Measurement of BitTorrent Swarms and their AS Topologies, Technical Report No. 464, November 2009

[16] The Mathworks: Matlab – The language of technical computing: http://www.mathworks.com/products/matlab/

[17] S. Le Blond, A. Legout, W. Dabbous, Pushing BitTorrent Locality to the Limits, INRIA, Dec. 2008