A stable random-contact algorithm for peer-to-peer file sharing

Hannu Reittu
VTT Technical Research Center of Finland, Hannu.Reittu@vtt.fi

***Abstract*—We consider a BitTorrent type file sharing algorithm with randomized chunk copying process. The system functions in completely distributed way without any 'Tracker', just relying on randomness. In such case the stability becomes an issue. It may happen, say, that some chunk becomes rare. This problem can persist and cause accumulation of peers in the system, resulting in unstable system. The considered algorithms result in processes similar to urn-processes. The rare chunk phenomenon corresponds to Polya-urn type process, where common chunks are favored. However, some urn-processes like the Friedman-urn can provide good balance by favoring rare chunks in copying process. Recently, we showed that an algorithm based on Friedman-urn is efficient in two chunk case. We generalize this algorithm for the more realistic case of many chunks. It shows good performance in terms of balance of chunks in an open system with constant flow of incoming peers. Further, the system is able to cope with instances like 'flash crowd', with large burst of incoming peers. The open system can also quickly reach equilibrium after an initial imbalance, when the system starts from a state with one rare chunk. We constructed a simplified model, assuming a good balance of chunks, and get results surprisingly close to simulations for Friedman-urn based random process.**

***Keywords*-file-sharing; urn-models; randomized algorithms**

I. Introduction

File sharing has been one of the first and the most popular application of peer-to-peer systems. The early applications like 'Napster' and 'Gnutella' were replaced by more advanced algorithms like BitTorrent, [1]. Such systems have shown capacity for large scale file distribution, [2]. These protocols are thus highly untrivial in performance and scalability, and are good motivation for interesting models to reveal what is their 'secret' of success. Further, such abstract models, if good enough, should also indicate ways to improve the protocol design. Here we report first results in this direction.

We consider a BitTorrent type system, which, however, does not contain any centralized elements like the 'Tracker' in real BitTorrent that controls who contacts who. Rather, our algorithm relies on randomness and is distributed. However, the simplest randomized algorithm leads to system that is similar to Polya-urn like system, with natural instability in open system setting with constant flow of new peers. In BitTorren the main innovation is that the file is divided into large number of smaller pieces or 'chunks'. Such chunks are copied from peer to peer. Since the chunks are small they are copied swiftly, which improves the performance, see also [3] for performance limits. In a randomized setting, it can easily happen that some chunks become rare, thus forming a bottleneck of performance. This happens because the most common chunks are easier to find, and, if no measures are taken, are favored in copying process. This is exactly what happens if we assume the simplest, Polya-type algorithms: peers make uniformly random contacts and copy what they find and after collecting all chunks departure.

Recently, we examined such a problem in a systems with just two chunks and the above problem of instability, [4], [5] was pinpointed. However, the two chunk case is unrealistic, because one should have many chunks to speed up copying. This is the issue of the current short paper. First we see that the same type of instability arises in the many chunk case as well.

The problem of stability in BitTorrent type systems has been studied also in [6] mostly with quite similar assumptions. However, the setting is different since the authors assume that the inflowing peers receive one uniformly random chunk upon arriving. The peers could obtain this chunk from the seed node that has all chunks. However, such a seed node becomes a server-like centralized element and possibly a bottleneck of performance. Then the system shows provable stability in fluid limit. We avoid the assumption of first random chunk, the peers arrive with no chunks and obtain every chunk from the network in a distributed manner. The untrivial result seems to be that the instability problem arises and can be, probably, avoided by a specific yet simple design.

We noticed, [5], that by modifying the random contact procedure to one that imitates the so called Friedman-urn (see e.g. [7]), the two chunk sharing process becomes remarkably stable and efficient. In this scheme, a peer that arrives does not have neither of the chunks, called chunk 0 and 1. The peers that have both chunks immediately leave

the system. As a result there are three types of peers in the system: peers without any chunks, and peers with chunk 0 or 1. There is also one peer called the seed, a permanent node with both chunks,which , acts as it has a random chunk, chosen independently for each time it is contacted. Peers arrive with constant rate λ.

The Friedman-urn process in two chunk case would mean that peers make uniformly random contacts to acquire missing chunks, if the target has chunk 0, it downloads the chunk 1. However, in our setting this is impossible since this chunk was not found. Our solution was ([5]) that the peer makes three simultaneous uniformly random contacts and downloads the chunk 0 if it sees the configuration $\{0,1,1\}$ and chunk 1 in case $\{0,0,1\}$, in other cases it does not download anything and makes another three contact trial until it does find such a configuration. The idea is that under the condition that a node succeeds to obtain a chunk, the probability that it downloads a particular chunk equals to that for the Friedman-urn process, in which the rarest chunk is favored. Such a system is able to cope with substantial imbalance, say, when the system starts in a state with one chunk very rare with respect to the other. Then, with constant rate of arrivals, the system quickly relaxes toward the equilibrium. Such a system in equilibrium shows also a good performance, the peers go through the system quickly.

II. Open system with random contacts

The two chunk case is not realistic, because the very idea of BitTorrent is to use many small chunks. So we have natural question: how to obtain stable system in such a case? If we have m chunks then we have $2^m - 1$, possible states of a peer. This is of course very large number, say, for $m = 100$, and the system is very complicated. In this situation, we generalized the algorithm as simply as possible. We use a slightly modified three random contact procedure, described in the Introduction.

The main points of our algorithm are: (i) All peers run the same procedure (ii)-(iii) independently of each other. (ii) A peer makes 3 simultaneous and uniformly random contacts with peers in the system. The peer that makes the contacts, learns which chunks those 3 contacts posses. Those chunks that only one contact has, are called the 'minority chunks'. The peer makes a list of those minority chunks found, that it self does not have. If this list is not empty, the peer downloads one of such minority chunk from the list, chosen randomly if there is more than one options. If the list is empty, the peer proceeds (iii). (iii) Repeat (ii) until all chunks are collected, then leave the system.

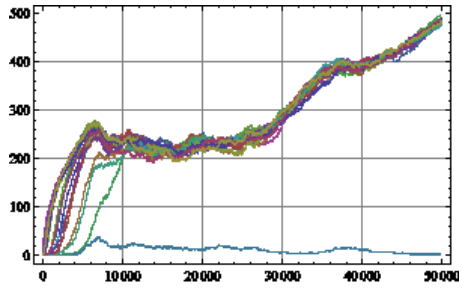

Figure 1. A result of a computer simulation with a simple random contact system, without favoring rare chunks. Each line represents the size of a particular chunk population, number of nodes in the system with given chunk, for system starting from a system with one 'seed' as a function of time (a sample path). Number of chunks $m = 20$ and $\lambda = 10$ times the contact rate. The populations of nodes that have certain chunks blows up, expect for one chunk that remains 'rare'.

Quite surprisingly, such approach seems to produce a stable and efficient system. Although not proven rigorously, simulations and simplified models seems to support this conjecture.

First we consider Polya-urn like, 'greedy' algorithm, in which a peer makes uniformly random contacts and downloads a missing chunk if it sees one. Then it repeats until it collects all chunks and departures. This seems to result in an inefficient and unstable system. One chunk becomes rare, and the number of peers keeps growing. This means that it takes longer and longer time for a peer to complete. This case is shown in Fig. 1.

This problem is persistent from case to case and is similar to the two chunk case, [5]. The other algorithm described above we call 'Forced-Friedman-algorithm'. As we can see, the results of simulation in Fig. 2 are promising. Indeed, the system shows very good performance and balance, peers go through the system almost with maximal possible rate, almost every contact is productive, the peer can find something to copy and moves ahead. This picture is also persistent from case to case.

Furthermore, it seems to have other good properties as well. Indeed, such a system seems to be able to cope with unstationar scenarios, a kind of 'flash crowds'. By this we mean that first there is a constant flow of incoming peers, but after some time this flow completely shuts down. If the system is unstable with poor balance of chunk populations,

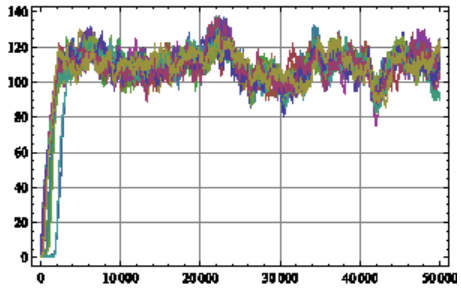

Figure 2. A computer experiment with random contact system with Forced-Friedman random contacts, $m = 20$, $\lambda = 100$. The sizes of chunk populations are shown starting from system with only the seed node. A good balance seem to prevail and a good performance, since no accumulation of peers is not seen although there is a constant flow of incoming peers into the system.

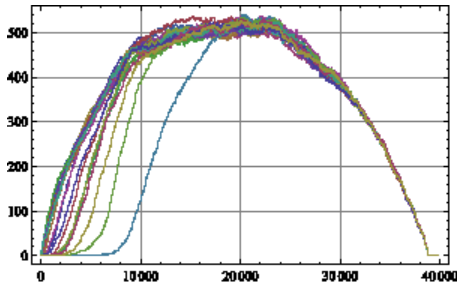

Figure 3. An unstationar scenario with Forced-Friedman algorithm. The systems starts from the empty state and with constant rate of incoming peers, then after a while the flow of peers stops. The case with 50 chunks, populations of nodes having particular chunks. The system manages to complete, all peers complete without any long tail of delay.

the system would not be able to complete, there would be a left-over, see also [8], [4]. The left-over is situation when some peers would not be able to complete (in system without seed) or would be forced to complete slowly by obtaining the last chunk from the seed. Our system seems to be able to avoid such difficulties as shown in Fig 3. Another good feature is the systems ability to cope with large initial unbalance. Even if there is a initially extremely rare chunk, the system quickly relaxes to steady state, as shown in Fig. 4.

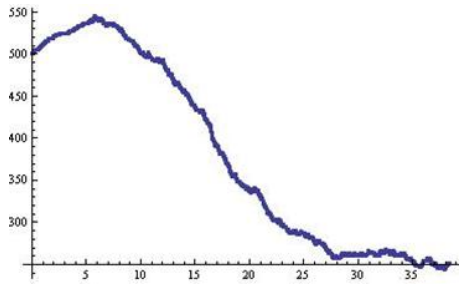

Figure 4. Relaxation of the system with Forced-Friedman algorithm. The system starts from a state with 500 nodes missing the same chunk (a rare chunk), however, the system quickly relaxes to steady state. A case with 20 chunks, number of peers in the system is shown.

III. A SIMPLE ANALYTICAL MODEL

The state-space of our system is enormous, so it seems to be impossible to create a useful model for this system. However, something can be done in this direction. Obviously some simplified assumptions must be done. We observed that the performance of the system is very close to ideal. By postulating this kind of behavior, a surprisingly accurate model can be found.

More precisely, we assume that the system is in an ideal state, meaning that all chunks are equally likely to be found in system. A peer that enters the system makes tree random contacts, and uses the Friedman-type logic to decide which chunk it can copy. If it founds at least one such chunk, it moves to state where it has one chunk, and so on. From this assumption, we deduce that probability that a particular chunk can be copied under the Friedman constrain is $3\frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{3}{8}$, let denote by $p = 1 - \frac{3}{8} = \frac{5}{8}$, the probability of the complement event. Then probability that a node with k missing chunks can copy a chunk in its current contact is $1-p^k$. Thus it is plausible to describe system by magnitudes $n_i, i = 1, 2, \cdots, m-1$, where n_i is number of nodes with i chunks. In the fluid limit one can assume the system of

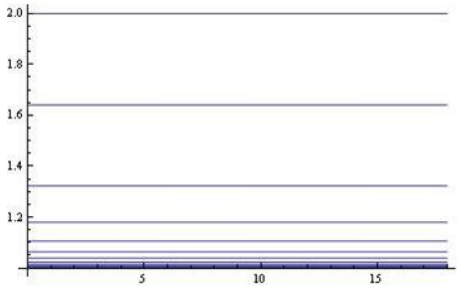

Figure 5. Levels of steady state population sizes of nodes with $1, 2, \cdots, m-1$ chunks in units of λ, the accumulation point of lines equals to 1.

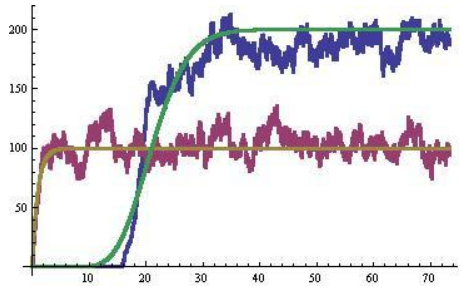

Figure 6. A simulation for 20 chunk-system with Forced-Friedman algorithm, $\lambda = 100$. The rugged lines are simulated processes for n_1 and n_{19}, while the smooth lines are solutions of the differential equations for the simplified model. Other components have similar behavior.

differential equations:

$$\begin{aligned}
\frac{d}{dt}n_1 &= \lambda - (1-p^{m-1})n_1 \\
\frac{d}{dt}n_2 &= (1-p^{m-1})n_1 - (1-p^{m-2})n_2 \\
\frac{d}{dt}n_3 &= (1-p^{m-2})n_2 - (1-p^{m-3})n_3 \\
&\cdots \\
\frac{d}{dt}n_{m-1} &= (1-p^2)n_{m-2} - \frac{1}{2}n_{m-1} \\
p &= \frac{5}{8}
\end{aligned}$$

They have the stationary solutions:

$$n_1 = \tfrac{\lambda}{1-p^{m-1}}, n_2 = \tfrac{\lambda}{1-p^{m-2}}, n_3 = \tfrac{\lambda}{1-p^{m-3}}, \cdots$$

$$\cdots, n_{m-2} = \tfrac{\lambda}{1-p^2}, n_{m-1} = 2\lambda.$$

The last relations mean that all populations have different sizes although they have an accumulation point $= \lambda$, as m grows, see Fig 5. These stationary solutions seems to be those that the real simulated system with Forced-Friedman algorithm yields, as shown in Fig. 6.

We made some computer experiments to see whether the empirical expectation is close to solutions of the simplified systems behavior, see Fig. 7. Based on those we conjecture that such means have damping oscillations around the curves of simplified model, and with very close to stationary level of population size. However, it can also be due some inaccuracy of the differential equations. Ideed, in the stochastic model the peer can point to itself and thus fail to download.

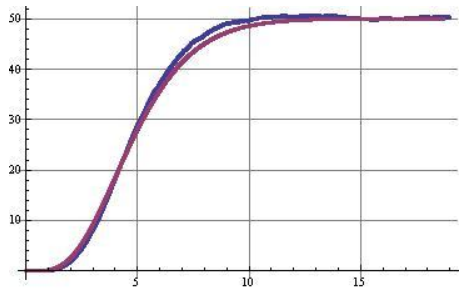

Figure 7. A simulation for 20 chunk-system with Forced-Friedman algorithm, $\lambda = 50$. A bit rugged line is simulated processes for n_5, empirical average over 1000 experiments, the smooth line is solution of the differential equations for the simplified model. It seems, that the average of the steady state is converging to the one for the simplified process. However, the transient state is slightly deviating from it, possibly having damping oscillations around the simplified system curve.

These simulations indicate also that the performance is almost ideal. This is because in the steady state, there is no 'bottleneck' chunks that are hard to find. That is why, there are few such contacts that do not lead to a download of a chunk.

IV. Conclusions

In this short paper we describe preliminary results on chunk copying system that relies entirely on randomness. Previously we studied two chunk case. It was shown that the system with Friedman-urn like algorithm is efficient and stable. In the case of many chunks we imitate this algorithm as far as possible. The resulting system shows stability and

good performance under dynamical conditions. Unlike the two chunk case, the simple proof based on Friedman-urn is not usable. The main challenge is to find rigorous foundation for this algorithm.

REFERENCES

[1] Cohen, B.: BitTorrent specification (2006) http://www.bittorrent.org.

[2] Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In: Proc. ACM Sigcomm, Portland, OR (2004)

[3] Mundinger, J., Weber, R., Weiss, G.: Analysis of peer-to-peer file dissemination. Performance Evaluation Review, Special Issue on MAMA 2006 (2006)

[4] Norros, I., Prabhu, B., Reittu, H.: Flash crowd in a file sharing system based on random encounters. In: Inter-Perf, Pisa, Italy (2006) http://www.inter-perf.org.

[5] Reittu, H., Norros, I.: Urn models and peer-to-peer file sharing. In: Proc. IEEE PHYSCOMNET'08, Berlin (2008)

[6] Massoulie, L., Vojnovic, M.: Coupon Replication Systems. In: Proc. ACM SIGMETRICS, Banff, Canada (2005)

[7] Pemantle, R.: A survey of random processes with reinforcement. Probability Surveys **4** (2007) 1–79

[8] Reittu, H., Norros, I.: Toward moldeling of a single file broadcasting in a closed network. In: Proceedings of IEEE SPASWIN2007, Limassol, Cyprus (2007)