Szemerédi-type clustering of peer-to-peer streaming system

Vesa Pehkonen and Hannu Reittu
VTT Technical Research Center of Finland
P.O. Box 1000, 02044 VTT
Finland
{Vesa.Pehkonen,Hannu.Reittu}@vtt.fi

***Abstract*—In this work we made a preliminary clustering analysis of an experimental peer-to-peer system, tested in a small scale experiments within PlanetLab. The application was an Internet-TV like streaming system based on Chord architecture. Our clustering is inspired by the Szemerédi's Regularity Lemma (SzRL). Such approach was already demonstrated in biology and appeared to be a powerful tool. Szemerédi's result suggests that the nodes of a large enough graph can be partitioned in few clusters in such a way that link distribution between most of the pairs look like random. Our main goal is to study what can this type of clustering tell us about p2p systems using our experimental system as source of data. We searched clusterings of Szemerédi-type by using max likelihood as guidance. Our graph is directed and weighted. The link direction indicates a client-server relation and the value is the proportion of all chunks obtained from such a link during the whole experiment. We think that the preliminary results are interesting. Most of the cluster pairs have very distinguished patterns of link distribution, indicating that such a novel approach has potential in classifying peers effectively. The values of weights between clusters and their distribution show some apparent patterns. We end up with 9 cluster pairs.**

Contributions: practical implementations of streaming system by V.P. and analysis by H.R.

I. Introduction

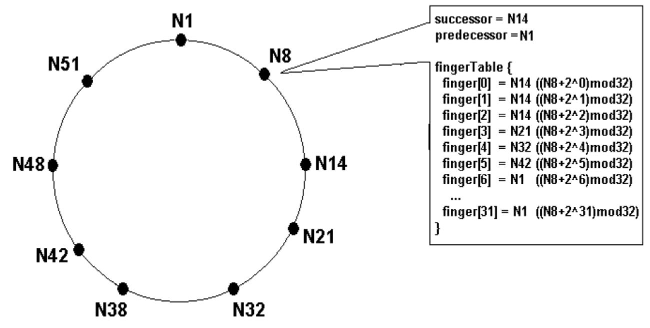
File sharing using the peer-to-peer techniques has shown to be very effective and popular among Internet users. Such systems like Bit-Torrent [1] and eDonkey are well known. These systems are typically used to obtain a file in small chunks that can be obtained in any order. A variant of this is streaming of the file, which means that the chunks are used on line by the application in a fixed order and without substantial delays, thus constraining the chunk downloading process to follow more or less a fixed order of chunks. The simplest form of streaming looks like watching a TV-channel. A peer that joins the network starts to obtain the file not from the beginning, but rather from the position available at that moment, and then proceeds in obtaining a stream of chunks and forwards an analogous stream, if requested, to other peers. Some such clients like the CoolStreaming, PPLive, SopCast, TVants and UUSee, (see Refs. in [2]) have already become popular for transmitting TV-channels through Internet.

For studying such streaming p2p systems we made an experimental client called 'PAN-STREAM' that was modified from our earlier experimental 'PAN-NET' client for fully distributed Bit-Torrent type file sharing, [3]. It is based on the Chord architecture, see Figure 1., and the corresponding experiments were run in PlanetLab environment, with around 50 peers. Chord has a ring topology with shortcuts called the fingers. As a first step we made a client that has basic functionalities, without trying to make it highly optimal. The results of first experiments were reported in [4]. We used a pull scheme, where the downloading peer can chose from whom it gets the next chunk. Then we made a bit more realistic scenario of a push model and made longer run of experiment. These new measurements are the data used in this work.

Application Seed/Leech
Streaming layer
TCP | UDP
IP

In the above table our protocol stack is shown. An application over peer-to-peer streaming layer is a simple command line or graphical user interface application whose main purpose is just to test functionalities of the system. There are two different types of peers: a seed and a leech. The seed reads the stream from file and sends it to the other nodes. The leeches download the stream from the seed and the other leeches. The streaming layer provides connections to the other nodes. Furthermore it provides to download and upload stream. The nodes communicate through TCP/IP. In normal case when a node leaves the network it is noticed, but if the node leaves badly it takes a few minutes before it is noticed. The latter case is not considered in this implementation. Furthermore there is no need for error checking of data transfer and the data packets are received in sending order. The UDP transfer is used only when a new node joins to the ring. The new node sends the FIND SUCCESSOR REQ message to the seed node. In this scenario the UDP transfer is used to avoid charging the seed node, because creating a TCP connection is a relatively heavy process.

There is one seed node that initiates the Chord network. When a leech node joins to the ring, it first connects to the seed to find a successor node and finger nodes. Then the node stays



```
successor = N14
predecessor =N1

fingerTable {
  finger[0]  = N14 ((N8+2^0)mod32)
  finger[1]  = N14 ((N8+2^1)mod32)
  finger[2]  = N14 ((N8+2^2)mod32)
  finger[3]  = N21 ((N8+2^3)mod32)
  finger[4]  = N32 ((N8+2^4)mod32)
  finger[5]  = N42 ((N8+2^5)mod32)
  finger[6]  = N1  ((N8+2^6)mod32)
   ...
  finger[31] = N1  ((N8+2^31)mod32)
}
```

Fig. 1. Chord network. Links or 'fingers' of one node, N8, are written. Only 'successor' links are shown.

waiting for the notify messages from other nodes. The nodes download chunks from their inverse fingers. A peer sends a message to all its antifinger neighbors informing them, which chunk it needs next. The uploading peer accepts those requests it can serve due to possible bandwidth restrictions. Each node has a chunk buffer that saves the newest downloaded chunks. The chunks are downloaded in numeral order. The peer that starts downloading a chunk informs its other neighbors that it does not need this chunk any more. Thus our experimental setting is unstructured and uses a push scheme for obtaining new chunks [2]. Unstructured means that we do not have a fixed delivery tree of chunks, rather it can change from chunk to chunk.

Our p2p network was designed as transparent and simple as possible to enable all kind of experiments and monitoring. It was not meant to be a practical client for some real applications.

A relevant question is how to classify peers in such systems based on their role in uploading and downloading. Ideally the system should be very homogeneous, a true peer-to-peer system, where every peer contributes to the systems resources equally. This would correspond to the case of just one cluster, the cluster of 'good peers'. However, this does not happen always. In an open system, say, there can be so called 'free-riders' that just download. A poor design can also result in such features. For systems performance, it can be useful to control and monitor peer classes. Thus, there is a task to classify or 'cluster' the peers based on their behavior. For BitTorrent a clustering was found usefull and peers with similar uploading capacities formed the clusters, [5]. However, this clustering analysis was based on prior hypothesis, such clusters were assumed based on the BT-algorithm. We would also like to see peers with similar behavior with respect to downloading and uploading characteristics in the same cluster, without any such prior hypothesis. We also do not assume that clusters are densely connected subgraph, as is customary.

Recently a clustering method inspired by the so called Szemerédi's Regularity Lemma (SzRL) in graph theory (for review see:[6]), was suggested and demonstrated in biology,[7]. It is a intriguing question to find out what is the significance of such clusters for p2p streaming? That is why we like to use same kind of approach adapted to directed and weighted graphs of our p2p streaming experiment. The arrows point from peer to uploader and the weight of a link is the ratio of chunks downloaded from a particular peer to the total number of chunks downloaded and both counted in the whole experiment duration.

Let us formulate SzRL for simple and undirected graphs. By $G(E, V)$, we denote a graph without loops or multiple links and with node set V and link set E. Let $X, Y \subseteq V$ be two disjoint subsets of vertices of G. The edge density of this pair of subsets is:

$$d(X,Y) = \frac{e(X,Y)}{|X|\,|Y|},$$

where $e(X, Y)$ is number of links in G with an endpoint in X and an endpoint in Y. By $|\bullet|$, we denoted the cardinality of a set. Further, a pair of vertex disjoint sets $A, B \subseteq V$is called ϵ-regular if for every $X \subseteq A$ and $Y \subseteq B$, such that $|X| > \epsilon\,|A|$ and $|Y| > \epsilon\,|B|$ we have

$$|d(X,Y) - d(A,B)| < \epsilon$$

A partition of vertex set V, into $k + 1$ sets, $\{V_0, V_1, V_2, \cdots, V_k\}$, where all except V_0, have equal cardinalities, is called ϵ-regular, iff all except at most ϵk^2 pairs are ϵ-regular.

Now we can formulate SzRL, [8]:

Szemerédi's Regularity Lemma: For every positive real ϵ and for any positive integer m, there are positive integers $N(\epsilon, m)$ and $M(\epsilon, m)$, such that any graph $G(E, V)$, with $n = |V| \geq N$ there is a ϵ-regular partition of V into $k + 1$ classes and $m \leq k \leq M$.

SzRL states, roughly speaking, that nodes of any large enough graph can be partitioned into k almost equal sized clusters, $1 \leq k \leq n$ (n is number of nodes) in such a way that almost all pairs of clusters look like random bipartite graphs with link probability equal to link density. The link density between clusters is the number of links joining the clusters, divided by product of cardinalities of corresponding clusters. This result is significant for large and dense graphs, when the link density is close to 1. However, a similar result is extendable also for sparse graphs with link density approaching 0 as $n \to \infty$, and even for real matrices, [9]. in weighted case, the link density is simply replaced by weight density and in sparse case, the ϵ at the right-hand side of regularity definition is multiplied be the link density of the entire graph.

It should be noted, that SzRL is true for all graphs starting from some lower bound for size, depending on accuracy, ϵ. However, the worst case scenario is such that this lower bound is enormous. Ideed for a given ϵ, the lower bound for graph size is like a tower of powers of 2:

$$2^{2^{\cdot^{\cdot^{2}}}},$$

where the tower has height that is upper bounded by $1/\epsilon^5$. Such a large numbers are too big to be considered in any applications! This has an important indication for an algorithmic version of the SzRL, introduced by Alon et al, [10]. Although the time complexity is only polynomial $O(n^{2.376})$, corresponding to the time for multiplying two $n \times n$ binary matrices, it requires this enormous size (n) of graph to be able to 'find' a regular partition. Fortunately, a considerable improvement was found in the recent work [11], where the running time is only linear in the graph size. Perhaps even more importantly, this randomized algorithm works in more realistic fashion, since it finds a regular partition for any graph or concludes that such partition does not exists. The algorithm works correctly with high probability.

Our graph is quite small and the SzRL is not literally applicable. However, the structure that is introduced by these results could be relevant as pointed out by [7]. Thus, we can not refer to SzRL as a guarantee of success, rather it just inspires to find such a regular like pseudo-random structures. This could be seen as a 'compression' of the graph: the clusters represent the structure while links between the clusters look like random and it is in these relations where the 'complexity' lies. This results in a kind of 'network of clusters' or 'reduced graph', where nodes are clusters and links between them have weights equal to link-densities between the clusters. Finding optimal clustering is in principle a very demanding task, because not only the clusters matter but everything is based on adjusting relations between clusters. Thus it is much heavier than usual clustering methods, that define clusters, say, as some well connected 'communities'. However, if succesfull such method can give very valuable information about the system. We use the max likelihood as a guidance, we try to extract a partitionining, where the pairs are as much as possible like random bipartite subgraphs. So, the task is not to fit the SzRL, but rather to find a similar structure in our graph. That is why we do not have parameters like ϵ or requirement that partitions should have equal sized classes. we simply try to do as well as possible, with criterion defined in next sections. Further, we try to analyze signicance of found structures and their potential for analyzing such systems. The regularity concept is by itself too vague and does not give us the exact picture of the relevance of SzRL-based clustering.

What we found were 9 cluster pairs and most of them look quite different from each other. Thus, on this qualitative level the method is successful since it can pinpoint different behavior patterns in our system. we think that a clear benefit of the method is that there is no prior assumptions what should the clusters look like and what kind of relations should they have between each other. In this sense, the method has potential of finding some unexpected features of the system.

II. Model description

Our task is to cluster nodes of a p2p system with $n = 48$ nodes. The experiment run for a time that corresponds to streaming of approximately 4000 chunks, and all except the seed get almost the same number of chunks during this period. The total (integral) number of chunks obtained by peer i is denoted as n_i. By $n_{i,j}$ we denote the integral number of chunks peer number i downloaded from peer number j. Then we define weighted graph by defining the weights for all pairs (i, j), $i \neq j$:

$$w_{i,j} = \frac{n_{i,j}}{n_i}.$$

Thus $0 \leq w_{i,j} \leq 1$. With this weighted and directed graph we associate a directed random graph $\mathcal{G}_w$ with independent links having probabilities

$$P((i, j) \in E_w) = w_{i,j},$$

which together with the link independence assumption defines the probability space. The reason why we defined $\mathcal{G}_w$ is technical. We assume that if the original graph has a Sz. structure, then $\mathcal{G}_w$ should have similar structure as well. A benefit is that for $\mathcal{G}_w$ we can use max likelihood fitting methods that allow computations. This results in some approximate scheme that is difficult to evaluate exactly. However, our guidance is mainly the result that we get, if the clusters are meaningful, the method is positively working. To define the Sz. structure framework we need two partitions $\mathcal{U} = \{U_1, U_2, \cdots, U_{Kout}\}$ and $\mathcal{V} = \{V_1, V_2, \cdots, V_{Kin}\}$ of node set V: $V = U_1 + U_2 + \cdots + U_{Kout}$ and $V = V_1 + V_2 + \cdots + V_{Kin}$, see in [7]. $\mathcal{U}$ is called out-group, and $\mathcal{V}$ in-group.

The Sz. structure in this case means that we have 'regularity' in link weight densities when we consider links going from an out-group to an in-group, taking into account all such pairs. As was said, this means that the link weight distribution for such pairs should be random-like with some concentration around the mean value. That is why we define the $Kout \times Kin$ matrix of weight density P as:

$$(P)_{i,j} = p_{i,j} = \frac{\sum_{\alpha \in U_i, \beta \in V_j} w_{\alpha,\beta}}{\mid U_i \mid\mid V_j \mid}.$$

We define yet another directed random graph, corresponding to P, $\mathcal{G}_P$, with independent links and with link probability:

$$P((i, j) \in E_P) = p_{u_i, v_j},$$

where u_i is defined from relation: $i \in U_{u_i}$ and, similarly v_j: $j \in V_{v_j}$. This latter graph is a kind of compressed version of the first one. $\mathcal{G}_P$ should be selected in such a way that it gives a Sz. type structure in a best possible way. This is a combinatorial optimization problem with huge search space, since the number of possible partitioning is big, even in our case.

We use a heuristic approach based on maximal expected log likelihood where we can use a greedy expectation maximization (EM) algorithm. The quality of approximation is then evaluated by using Akaike information criteria (AIC), with minimal value giving the best choice of parameters. Thus our approach is quite similar to that in [7] where a binary directed graph was considered. The main emphasis of [7] was in predicting unknown connections based on a partially observed network. It seems that such a method can be very successful in this type of tasks. We are just interested in clusters themselves and link patterns between them. Thus the task is to structure the p2p network. However, due to computational difficulties the method is probably restricted to moderate size graps like ours. On the other hand, for a very large graph, the algorithmic version of SzRL, [10] could be used, as was already done in [12] in the digital image segmentation. Such an algorithm that finds clusters indicated by SzRL runs in polynomial time with respect to the number of nodes. Probably even better algorithm would be the one sugessted in [11], that can be used even for small graphs. Unfortunately, we noticed this paper only after this paper was almost finished. However, we are looking forward applying such algorithm to find and compare Sz. clusterings in our case.

The goal function, expected log likelihood, that we want to maximize is

$$l(P) = E_{\mathcal{G}_w} \log P(G \in \mathcal{G}_P), \tag{1}$$

that is, by definition:

$$l(P) = \sum_{1 \le i,j \le n} \left(w_{i,j} \log p_{u_i,v_j} + (1 - w_{i,j}) \log(1 - p_{u_i,v_j}) \right). \tag{2}$$

The AIC index is defined as

$$AIC = -2 \max_P l(P) + 2K,$$

where K is the number of parameters of the model. in our case $K = Kin \times Kout + 2n + 2$, the first term defines the dimension of probability matrix P, the second corresponds to two functions that define two partitions and number 2 corresponds to Kin and $Kout$. That is why we have:

$$AIC = -2 \max_P l(P) + 2Kin \times Kout + 4n + 4. \tag{3}$$

III. Clustering analysis

We used a greedy EM-algorithm similar to that used in [7]. For fixed $Kout$ and Kin, the algorithm starts from random clusters. Then the P-matrix is calculated. Then for each peer the algorithm finds clusters in such a way that the expected log-likelihood function $l(P)$ is maximal. This is the greedy step of algorithm and the maximisazion of likelihood is done only by changing position of peers in the clusters one-by-one. This is necessary since the proper maximation of likelihood in just one step is practically impossible. However, for EM-algorithms it is known that even this kind of incremental optimisazion finds at least a local maximum [13]. Then the algorithm was run several hundred times to capture the global maximum log-likelihoods. The experiments were done using all combinations of $(Kin, Kout)$ in the range of both in

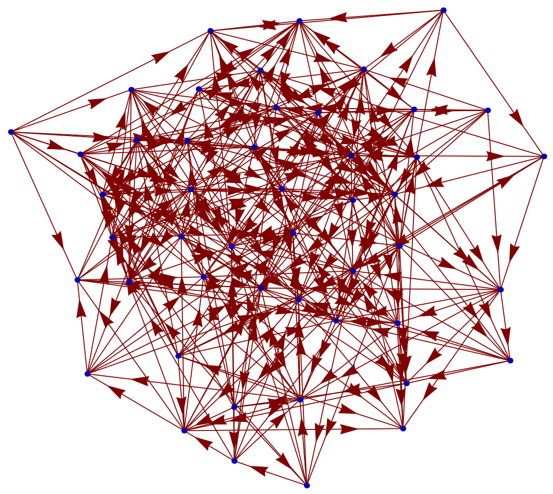


Fig. 2. 48 nodes representing peers in our PlanetLab experiment and directed links pointing from downloading peer to uploading one

$(2, 3, \cdots, 10)$, and AIC parameters were calculated. The case of no clusters was considered also. This would be an Erdös-type random graph with link probability equal to the link density of the entire network. However, in our case we must take into account that some links are 'impossible' because only antifingers are used. That is why we used a restricted random graph where all such possible links were considered with equal probability, and the AIC was calculated for such a model. It turned out that the optimum was $Kin = 3, Kout = 3$, although the minimum was not very sharp. Some close values could be also considered as plausible ones, but we prefer a symmetric case with $Kin = Kout = 3$. For larger graphs one could use spectral methods to define the range of possible cluster numbers, [7], and thus reducing the computations needed.

Figure 2 shows the original graph to be clustered. This picture is of cource very confusing, and the need for some kind of preprocessing is quite obvious.

In Figure 3, the histogram of non-zero link weights of the whole system is shown. It has a broad peak around mean value. There is also a peak for very small weights. This could be interpreted as a noise. When the system starts, the fingers can change their targets, and meanwhile few chunks are downloaded from these antifingers.

As a result of our Sz. clustering, we get a reduced graph where the nodes are in- and out-groups, see Figure 4. Link weights are average link weights between those clusters. Heavily connected pairs are the ones that are important. However, since the corresponding in- and out groups overlap, it is difficult to see in details what are these important pairs like.

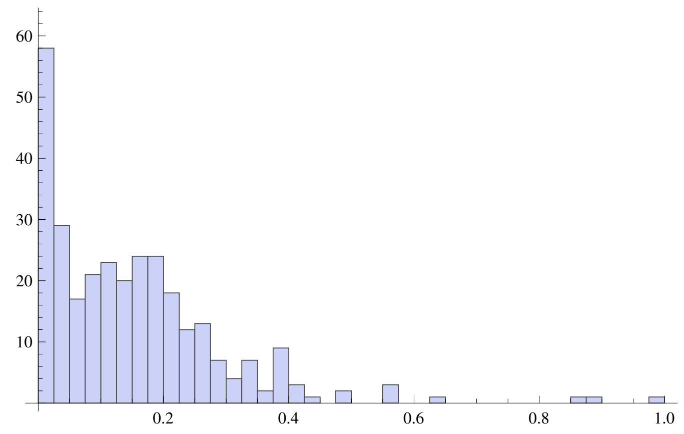


Fig. 3. Histogram of all non-zero weights in the peer-to-peer network. Mean value is 0.152061

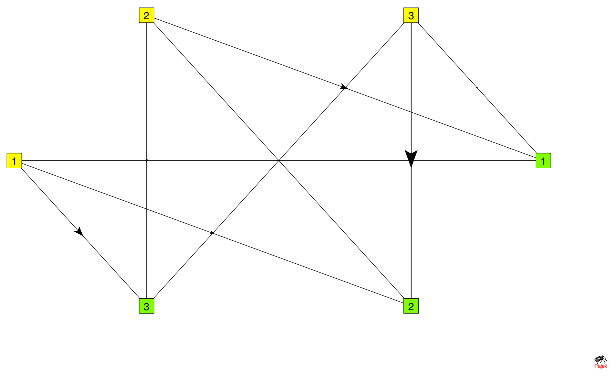


Fig. 4. Relations betwee 3 out - and 3 in groups or the 'reduced graph' of the original graph. The line thickness is proportional to mean weight between pairs.

That is why we draw characteristic examples of those pairs. We used the following visualization scheme for all graphs. For each pair we get 4 groups, a kind of 'bow-tie'. First are nodes that have only outgoing links, second those that have only incoming links, third those that have both and finally those that are outside the pair is a link-less circle. In these terms we found very different patterns, which is of cource what we were after.

The sizes of clusters were not equal but there were not very small clusters either. The non-zero weight distribution between 9 pairs seems to show indeed qualitatively different distributions, with different mean values. Thus, although our experiment was small scale and the streaming algorithm quite basic, the clustering method seems to be promising. In the following Figures, we show some subgraphs that also support this view.

In Figure 5 is shown the pair $(1,1)$, which has very weak links, as can be seen already from the reduced graph in Figure

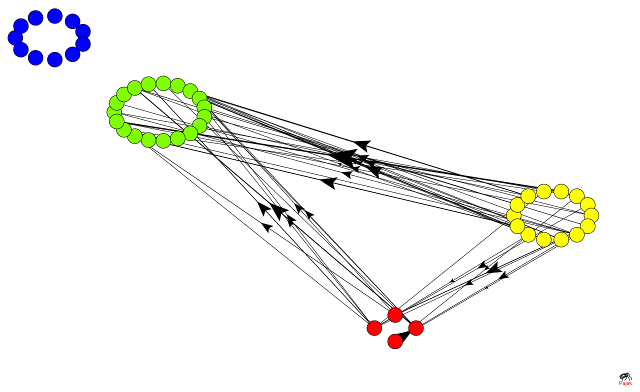


Fig. 5. The first cluster pair $(1,1)$. The link weights are low. The smallest group with 4 red nodes is the one where both in and out links exist, the yellow group has only outgoing links and finally, green nodes have only incoming links. Blue nodes are outside these two clusters. This is almost like: 'the yellow nodes download a little from the green nodes'.

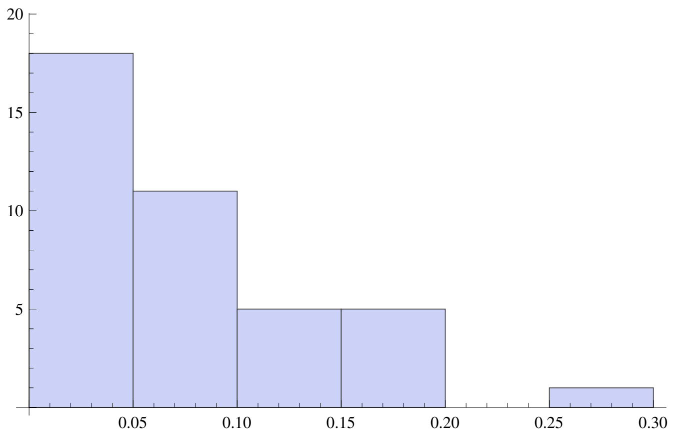


Fig. 6. Histogram of the distribution of non-zero weights for the $(1,1)$ cluster pair. Mean value is 0.0722, with small link weights prevailing.

4.

The histogram of weight distributions of pair $(1,1)$ is shown in Figure 6. Qualitatively it is similar to exponential distribution, with very low mean.

In Figure 7 we show an example of important pair $(1,3)$, with high level of weights. In this case all 3 parts of bow-tie are similar in size, forming a kind of regular triangle.

In Figure 8 we show the histogram of weights for pair $(1,3)$, a slight peak is apparent around the mean value.

For pair $(1,3)$ we draw weights for all peers involved, see Figure 9. Each line corresponds to a peer in this cluster pair. The lines gives the link weights of the corresponding peer in this cluster pair listed in decreasing order. The values are read at integer x-coordinates and are joined with lines for clarity. The lines are not very flat, as should be. However, due to limited size of network this could well be the best one can

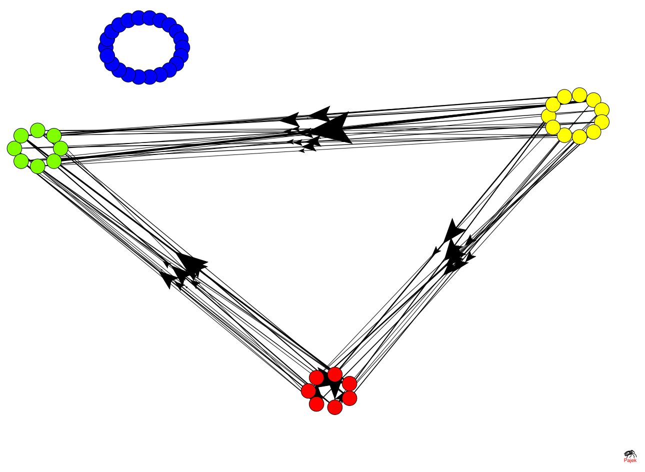


Fig. 7. Another cluster pair $(1, 3)$, with high weights. Here all tree groups are similar in size.

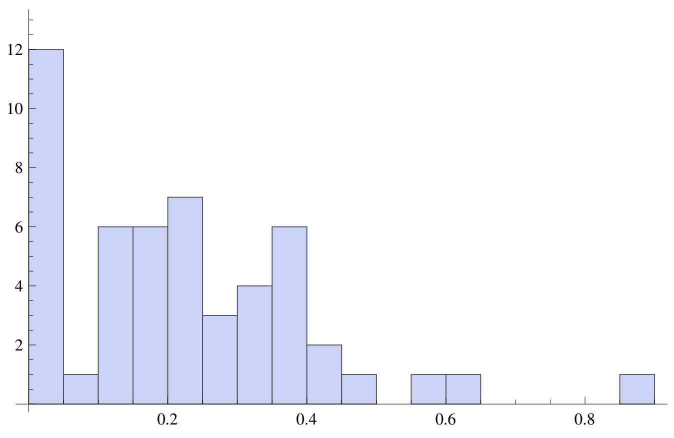


Fig. 8. Histogram of distribution of non-zero weights for $(1, 3)$ cluster pair, with mean 0.22319.

hope. In any case this pair has some high weight levels, that was correctly detected by Sz. clustering.

The importance of cluster pair $(1, 3)$, is seen in the plot of Figure 10, where total weights of peers within these clusters are shown, each dot represn the value of its total weight, counted in this cluster pair. It shows that all peers get a substantial part, typically more than 50% of all chunks, in relations within these clusters.

Next example, the pair $(3, 1)$ are clusters that have very low link weights between them, see Figure 11.

Its distribution of weights is almost a constant small value, shown in Fig. 12.

Yet another quite important pair is $(3, 2)$, see Fig. 13, which is like 'client-server', where on party downloads a lot from the other one.

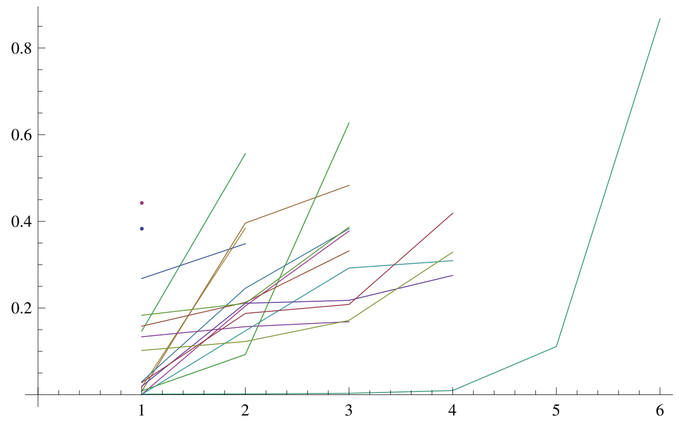


Fig. 9. Link weights for $(1, 3)$ cluster pair. Each line represents link weight for each peer .

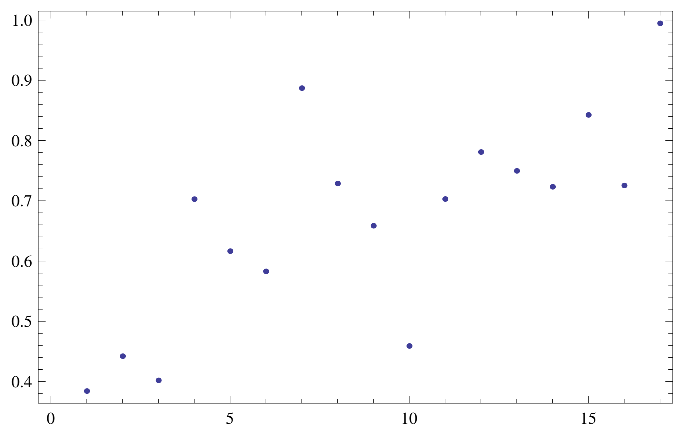


Fig. 10. Integral link weights for cluster pair $(1, 3)$, high values are indicators of importance of this substructure. A substantial portion of chunks are copied within it.

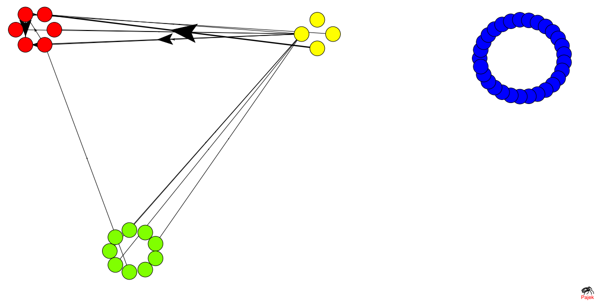


Fig. 11. This pair $(3, 1)$ of clusters is with only modest link weights, these clusters are almost independent of each other. It is an example of a cluster pair with almost zero weight density.

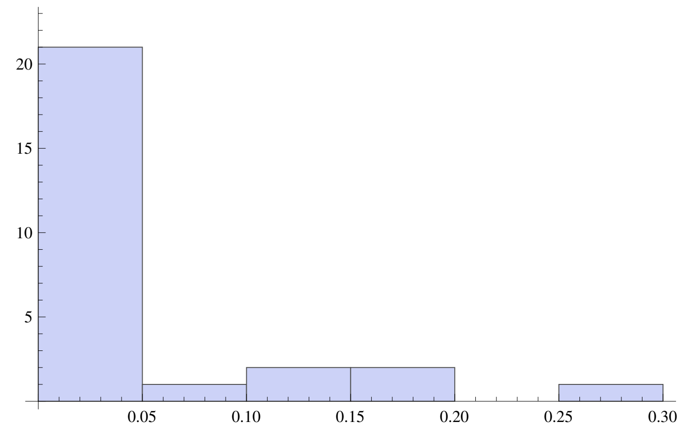


Fig. 12. Histogram of distribution of non-zero weights for $(3, 1)$ cluster pair. It is almost just one small weight value

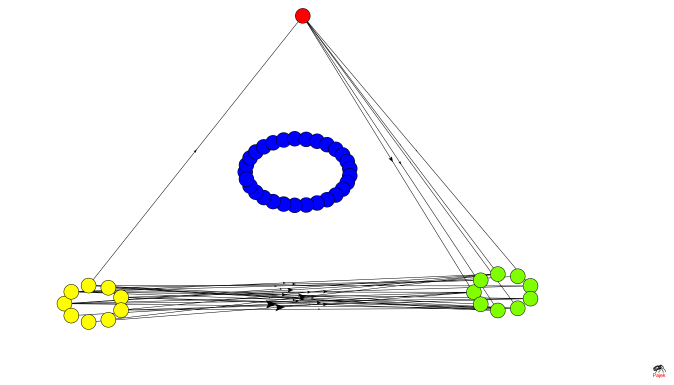


Fig. 13. This pair $(3, 2)$ is with high link values. It is mostly of 'client-server' type, one group is downoalding from the other.

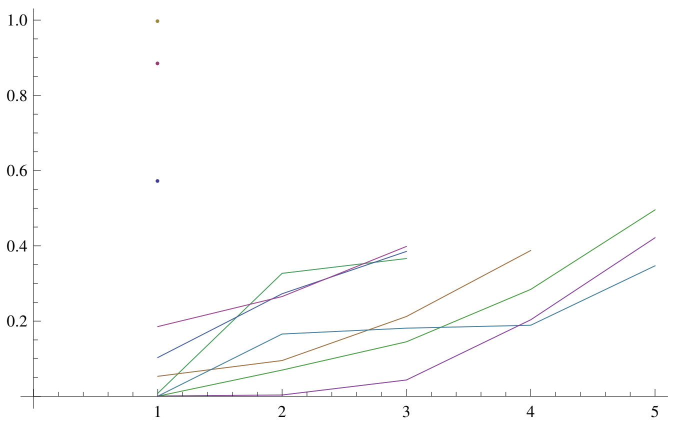


Fig. 14. Link weights for pair $(3, 2)$. Each line represent link weights for one peer, the values are read from integer coordinates. Link weights are large and quite uniform with flat profile.

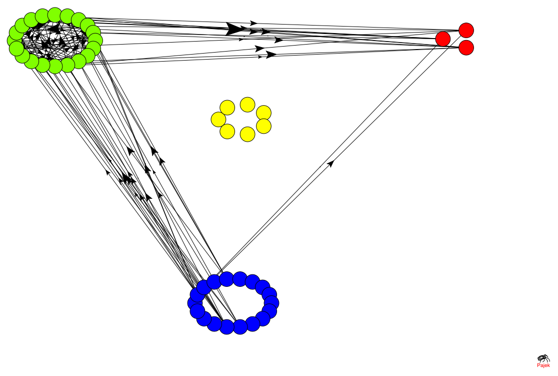


Fig. 15. Subgraph of clusters $(2, 1)$. Notably there is a set of peers that are both uploading and downloading much.We can say that the subgroup of green nodes is a 'good' group in sense that there is good level of balance between uploading and downloading.

The downloading profiles of all peers in pair $(3, 2)$ are shown in Fig. 14.

An interesting pair is $(2, 1)$, with 'peer-to-peer'-like heavy weight clusters, see Fig. 15. This means that there is much reciprocity in links, many peers both upload and download a lot within this pair.

The histogram of weights in pair is shown in Fig. 16. It has a broad peak around the mean value $0.148...$.

In this case we get quite flat link weights for peers in $(2, 1)$, see Fig. 17.

The integral weights of pair $(2, 1)$ are shown in Fig 18, indicating that this pair is important for streaming, because most of the chunks of peers involved are obtained within this pair.

IV. Conclusion

This preliminary analysis gives us some hope that Szemerédi-type clustering approach could give some significant information about a p2p system and reveal some statistical

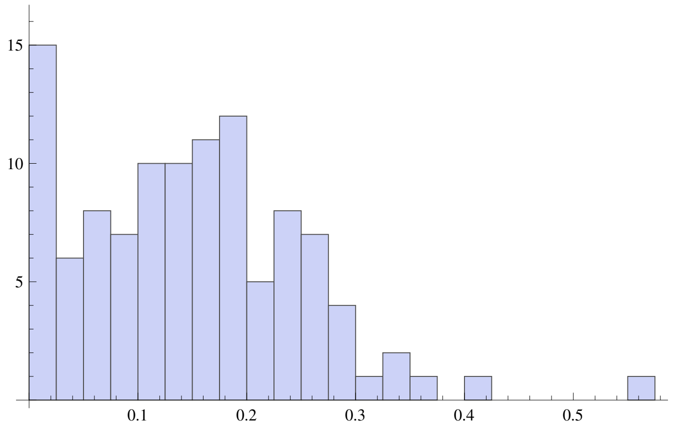


Fig. 16. Histogram of distribution of non-zero weights for $(2, 1)$ cluster pair.

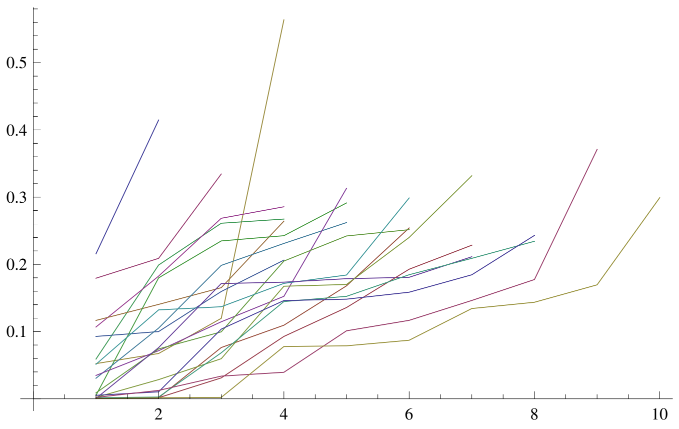


Fig. 17. Distribution of weights on links for clusters $(2, 1)$. Each line represents link weights for each peer in cluster 2 pointing to cluster 1. Most lines are quite flat indicating uniformity.

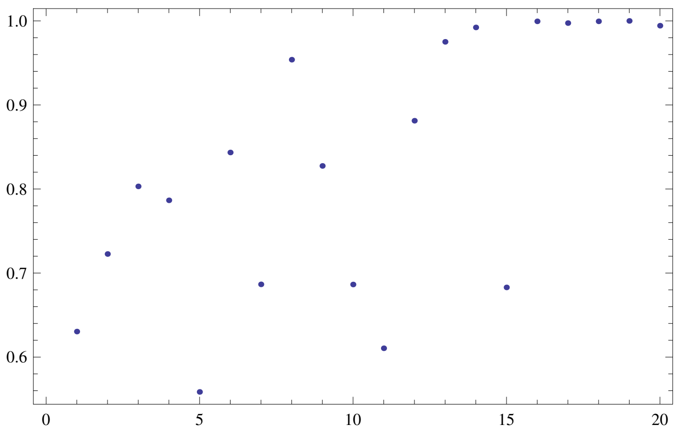


Fig. 18. Integral weights of peers within clusters $(2, 1)$. Each point gives the proportion of chunks obtained within this subgraph. High level of these magnitudes indicate that this is an important subgraph in the peer-to-peer network.

properties of peers' behavior, otherwise almost invisible. This happened although our system was very small, only with 48 nodes. For larger systems one could expect sharper results, in the sense that weight distributions for cluster pairs would be more concentrated around the means, thus making the pairs 'more regular'. The clustering was found as a result of combinatorial optimization, which means a huge search space in contrast to traditional clusterings that are easier to find. As a result the Sz. clustering can also give some untrivial information about the system that is difficult to reveal without such a systematic approach.

Our system was quite small, yet it was large enough to prevent the use of direct exhaustive methods to find clusters. Instead we used a greedy algorithm and run it several times to find the global optimum. In more thorough study one should use approaches like Monte Carlo simulations, [7], to find reliably the global optimum. For larger p2p systems algorithmic version of SzRL could be more plausible solution. Further challenges would be finding a distributed algorithm that can find the Sz. type clusterings. In 'usual' clustering where clusters are associated with dense subgraphs methods based on random walks were suggested, [14]. An interesting suggestion was that the regular partitioning can be explicitly found by generating random neighborhoods of graph, [15]. This could be a direction to find a distributed Sz. clustering algorithm.

Acknowledgments This work was supported by TEKES as a part of the Future Internet program of TIVIT. We would like to thank prof. Ilkka Norros for reading the manuscript. We also thank anonymous referees for valuable comments.

References

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

[2] Bonald, T., Massoulie', L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: Optimal performance trade-offs. In: Proc. SIGMETRICS'08, Annapolis, Maryland, USA (2008)

[3] Norros, I., Pehkonen, V., Reittu, H., Binzenhofer, A., Tutschku, K.: Relying on randomness - PlanetLab experiments with distributed file-sharing protocols. In: Next Generation Internet Networks 3rd EuroNGI Conference, Trondheim, Norway (2007)

[4] Norros, I., Pehkonen, V., Reittu, H.: Peer-to-peer streaming experiments based on chord overlay. In: Euro-NF International Workshop on Traffic and Congestion Control for the Future InternetNext Generation Internet,EuroNF-TCCFI, Volos, Greece (2011)

[5] Legout, A., Liogas, N., Kohler, E.: Clustering and sharing in BitTorrent systems. In: Proc. SIGMETRICS'07, California USA. (2007)

[6] Komlós, J., Simonovits, M.: Szemerédi's regularity lemma and its applications in graph theory, in: D. Miklós and V.T. Sós and T. Szonyi, editors, Combinatorics, Paul Erdos is Eighty. János Bolyai Mathematical Society (1996) Budapest, pp.295-352.

[7] Nepusz, T., Négyessy, L., Tusnády, G., Bazsó, F.: Reconstructing cortial networks: case of directed graphs with high level of reciprocity in: B. Bollobás and R. Kozma and D. Miklós, editors, Handbook of Large-Scale Random Networks, volume 18 of Bolyai Society of Mathematical Studies. Springer (2008) pp. 325-368.

[8] Szemerédi, E.: Regular partitions of graphs. In: CNRS, Paris, pp. 399-401 (1978)

[9] Scott, A.: Szemerédi's regularity lemma for matrices and sparce graphs. (Combinatorics, Probability and Computing) to appear.

[10] Alon, N., Duke, R., Lefmann, H., Rödl, V., Yuster, R.: The algorithmic aspects of the regularity lemma. Journal of Algorithms **16** (1994)

[11] Fischer, E., Matsliah, A., Shapira, A.: Approximate hypergraph partitioning and applications. (SIAM Journal on Computing (SICOMP) (to appear))

[12] A.Sperotto, Pelillo, M.: Szemerédi's regularity lemma and its application to pairwise clustering and segmentation. In: Proc. EMMCVPR, LNCS 4679, Ezhou, China (2007)

[13] Neal, R., Hinton, G.: A view of the EM algorithm that justifies incremental, sparse, and other variants . In: Learning in Graphical Models (M.I. Jordan, ed.), MIT Press (1998)

[14] Latapy, M., Pons, P.: Computing communities in large networks using random walks. J. Graph Algorithm Appl. **10** (2006)

[15] Tao, T.: Szemeredis regularity lemma via random partitions (2011) http://terrytao.wordpress.com/2009/04/26/szemeredis-regularity-lemma-via-random-partitions/.