Comparison of Piece-Picking Algorithms for Layered Video Content in Peer-to-Peer Networks

Michael Eberhard*, Riccardo Petrocco†, Hermann Hellwagner*, and Christian Timmerer*
*Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria, firstname.lastname@itec.aau.at
†Technische Universiteit Delft, Delft, The Netherlands, r.petrocco@gmail.com

Abstract—**The distribution of video content over P2P systems has become a popular and cost-effective option in recent years due to the increasing online availability of content. Additionally, the diversity of end-user terminals used for consuming content demands the provisioning of content in different qualities. Both problems are addressed by distributing layered video content over P2P networks. A piece-picking algorithm for layered video content needs to ensure that the pieces are received in time as well as that the best possible quality that can be processed by the end-user terminal and downloaded under the given network conditions is provided. In this paper, we describe our algorithm for piece-picking of layered video content in Bittorrent-based P2P systems and compare it to other existing piece-picking algorithms. The evaluation presented in this paper shows that our algorithm can very well compete with previously published algorithms.**

I. Introduction

Today, more and more multimedia content is distributed online in high quality. As the number of users of online multimedia portals is quickly increasing, the distribution costs are rising as well. P2P systems provide a cost-effective and scalable alternative, since the users are distributing the content to other users while consuming it.

Furthermore, multimedia content is nowadays consumed on a variety of different end-user terminals with diverse capabilities. These terminals include high-definition TV-sets, laptops, and also a number of mobile devices like smartphones or tablets. Thus, video content needs to be provided in multiple qualities to ensure that the content can be consumed on all these devices in appropriate ways. While it is possible to encode a piece of content multiple times and provide the right version to the user based on the user terminal's capabilities, this is not the best solution for P2P systems since in that case only users consuming the same video in the same quality could share the content.

Layered video codecs like the Scalable Video Coding (SVC) extensions [1] of the Advanced Video Coding (AVC) standard enable providing multiple video qualities within a single video bitstream. While it is sufficient to only receive the base layer (the first quality layer) for playback, all other layers that are received in time improve the playback quality. The usage of layered video codecs is especially interesting for P2P systems since all users consuming the same content can share at least the base quality layer, and any two users can share additional layers up to the highest layer that both are consuming.

In this paper, a piece-picking algorithm for layered video content in Bittorrent-based P2P systems is presented and compared to other piece-picking algorithms. In a P2P system, the piece-picking algorithm decides which pieces are downloaded at which point in time. The algorithm works on a sliding window which contains the pieces for all layers starting at the current playback position into the near future. When downloading layered content, the algorithm needs to find a good balance between three requirements: continuous playback, best possible quality, and avoidance of frequent quality switches.

To ensure a smooth continuous playback, the piece-picking algorithm needs to avoid frame skipping, which can be achieved by prioritization of base layer pieces. When trying to provide the best possible quality the algorithm needs to prioritize higher layers with close playback deadlines over lower layer pieces with a later playback deadline, to ensure that the higher layers which improve the playback quality are received in time. This second requirement is often in conflict with the first requirement, continuous playback. The third requirement, avoidance of frequent quality switches, causes the algorithm to switch only to a higher layer (and better playback quality) if the new layer can be kept for some time. The reason for this requirement is that frequent switching of the playback quality is disturbing for the user [2]. The consideration of all three requirements presents a challenging optimization problem that needs to be solved by the piece-picking algorithm.

To evaluate our piece-picking algorithm, it has been integrated into the open-source Next-Share P2P system [3]. Additionally, other piece-picking algorithms have been integrated into NextShare as well in order to enable us to evaluate how our piece-picking algorithm compares to others in terms of performance. The source code of the NextShare P2P system including the implementation for layered content is available at [4].

The remainder of this paper is organized as follows. In Section II, previously published related work is discussed. In Section III, our piece-picking algorithm for layered content is described. Section IV describes the evaluation results of our real system tests. Finally, Section V concludes the paper.

II. Related Work

The distribution of layered content has been a popular topic in recent years. [5] describes a very well defined solution for integrating layered content into a P2P system. However, the functionality of the piece-picking algorithm is not described in detail. Although the solution has been integrated into a P2P client, the client has been specifically designed for the distribution of layered content and only AVC codecs have been used for the implementation (which support only temporal scalability).

[6] describes a piece-picking algorithm for layered content, but the proposed architectural choice impedes the algorithm's easy integration into existing Bittorrent systems. PALS [7] describes a receiver-based algorithm that allows to download the desired quality. However, the quality dimensions of SVC are not specifically addressed in the solution and an integration into existing Bittorrent systems would again be difficult.

Our piece-picking algorithm has specifically been designed to take the requirements of the Bittorrent protocol (e.g., fixed piece size) into account in order to enable easy integration into existing clients and backwards compatibility (clients not supporting SVC can still download and share the SVC base layer). Additionally, the algorithm has been integrated into the open-source Bittorent-based NextShare P2P system.

III. Piece-Picking Algorithm

The task of the piece-picking algorithm in Bittorrent-based P2P systems is to decide which pieces should be selected for download at which point in time. To optimize the playback quality for the user, the piece-picking algorithm needs to take the three factors mentioned in Section I, avoidance of frame skipping/stalling, providing the best possible quality, and avoiding frequent quality switches, into account.

The formal framework for our algorithm has already been presented in [8]. A brief description of our algorithm is given in the following. The algorithm is based on algorithms for the solution of the Knapsack Problem (KP) [9], which is a problem in combinatorial optimization similar to the piece-picking problem. The algorithm applies a greedy approach, where the utility for each of the pieces within the sliding window is calculated and as many pieces as possible are downloaded within the available bandwidth. To avoid frequent quality switches, the algorithm only performs quality switches to higher layers if the pieces of the new layer have already been downloaded for multiple future time slots. This algorithms is from now on referred to as the KP algorithm.

The utility u_{ijk} of a piece p_{ij} with the playback time slot t_i and layer l_j is calculated using the following formula.

$$u_{ijk} = \frac{d_j \times dp_{ijk}}{(t_i - t_k)^{\alpha}} \quad (1)$$

The distortion reduction importance d_j of a piece defines the importance of a layer, where lower layers have a higher distortion reduction importance than higher layers. The download probability dp_{ijk} is the probability that the piece is received in time for its playback time slot and that the piece is useful and can be decoded (i.e., all pieces of the same time slot and lower layers are also received in time). t_k specifies the decision time point at which the utility for the piece is calculated.

To calculate the utility, the distortion reduction importance d_j is multiplied with the download probability dp_{ijk}. This

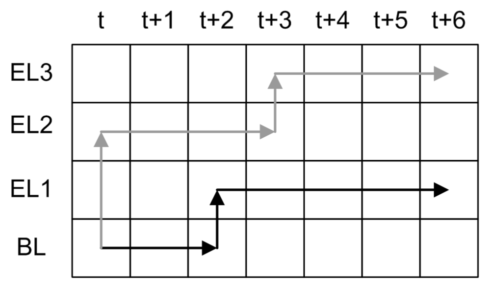

Fig. 1: KP Algorithm

ensures that higher layer pieces are only selected for download if there is still sufficient time until the playback deadline to download the piece and all lower layer pieces. The result is subsequently divided by the remaining time slots until playback ($t_i - t_k$), to ensure that urgent pieces that will be played back soon are downloaded with additional priority. The urgency weighting α is utilized to influence the importance of the distortion reduction compared to the remaining time slots. By changing the urgency weighting, the trade-off between always downloading the best possible quality (high α value) and ensuring continuous playback without stalling or frame skipping (low α value) can be adjusted.

The setting of the parameters of the KP algorithm is based on the bandwidth conditions and is determined after the initialization phase and possibly at certain points during the streaming process. If, e.g., the bandwidth conditions are critical, a low urgency weighting and/or a higher distortion reduction importance of the base layer are applied to ensure continuous playback.

An example of how the KP algorithm works is illustrated in Figure 1, where the Y axis represents the layers, the X axis represents the time slots of the sliding window, and each cell represents one piece. The KP algorithm does not follow a predefined scheme for piece-picking, but calculates the utility for each piece in the sliding window and bases the piece selection on the remaining download time and the bandwidth conditions. Thus, the piece-picking of the KP algorithm could follow, e.g., the lower line if the bandwidth conditions do not allow downloading many layers. On the other hand, the KP algorithm could follow the higher line if the bandwidth conditions are good.

IV. Evaluation

In this section, our piece-picking algorithm for layered content is evaluated and compared to other piece-picking algorithms. For the evaluation, the piece-picking algorithms have been integrated into the NextShare P2P system. The KP algorithm described in Section III is compared to two other algorithms, the baseline algorithm and the zigzag algorithm.

The baseline algorithm [3] prioritizes the pieces within the sliding window based on layers only and is illustrated in Figure 2. The baseline algorithm selects all pieces of the base layer according to their deadlines first, then all pieces from the first enhancement layer, then from the second enhancement layer, and so on. By applying this strategy, the algorithm ensures

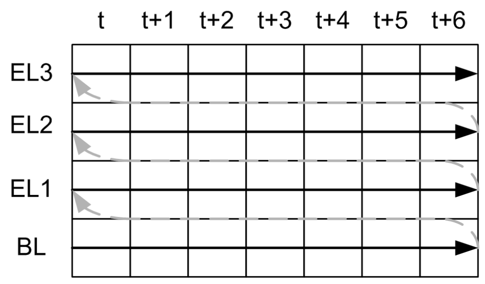

Fig. 2: Baseline Algorithm

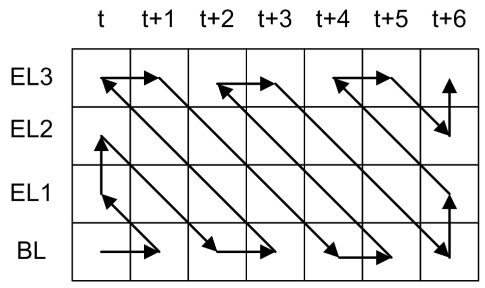

Fig. 3: Zigzag Algorithm

that frame skipping is minimized since base layer pieces are always prioritized, and that only useful pieces are downloaded since pieces of an enhancement layer cannot be selected for download before all lower layer pieces are selected.

The zigzag algorithm [6] downloads the pieces based on a zigzag priority allocation and is illustrated in Figure 3. The zigzag algorithm prioritizes lower layer pieces, but higher layer pieces with close playback deadlines are more important than lower layer pieces with a later playback deadline. This allows to increase the playback quality faster than in the baseline algorithm, but might lead to frequent quality switches.

The piece-picking algorithms were evaluated in a test lab with Linux machines. The test lab consists of server machines, on which the seeding peers are running, and client machines, on which the peers that stream the content are running. Between the server and client machines, routers (Linux machines as well) are utilized which allow to influence the network conditions between the peers. The routing machines use the Netem [10] kernel component to emulate network characteristics like delays or upload and download bandwidths. By changing the parameters of the routers, the following four scenarios could be evaluated for the piece-picking algorithms:

- *Scenario 1*: In this scenario, the seeders provide more than the bandwidth required for the download of all layers to the clients. This scenario illustrates the behavior of the algorithms under optimal bandwidth conditions.
- *Scenario 2*: In this scenario, the bandwidth from the seeders is strongly limited to provide slightly less bandwidth than required for the playback of the first two layers. This illustrates the behavior of the algorithms under critical bandwidth conditions.
- *Scenario 3*: In this scenario, the seeders provide slightly more bandwidth than required for streaming three layers to the peers.
- *Scenario 4*: This scenario uses the same parameters as Scenario 3, except that a 5% churn rate is applied, i.e., each seeding peer has a chance of 5% at every time slot to leave the swarm and join again afterwards.

TABLE I: Layer Structure

Layer	**Bit Rate** [kbit/s]	**Resolution**
BL	512	480p
EL1	1024	480p
EL2	1536	480p
EL3	2048	480p

TABLE II: Parameter Settings

Parameter	**Value**
Video Duration	37 min
Piece Size	168 KB
Time Slot Size	2.56 s (64 frames)
Sliding Window Size	10 time slots
Initialization Phase	2 time slots
Seeders Upload Capacity Scn. 1	3000 kbit/s
Seeders Upload Capacity Scn. 2	800 kbit/s
Seeders Upload Capacity Scn. 3/4	1800 kbit/s

The reason for defining these four scenarios was to evaluate the piece-picking algorithms under optimal bandwidth conditions, critical bandwidth conditions, and bandwidth conditions that trigger frequent quality changes.

For each scenario, 100 peers are streaming the video sequence *Big Buck Bunny* (concatenated three times to reach the desired length) that is encoded with the layer structure presented in Table I. Each layer has a bit rate of 512 kbit/s, the cumulative bit rate for each layer is shown in the table. The parameter settings for all four scenarios are presented in Table II.

Before streaming of the layered content starts, the sliding window is initially filled during the initialization phase. This ensures that the playback is not immediately stalled at the beginning of the streaming process if a base layer piece is not received in time.

To compare the three algorithms, the average received bit rate of the peers for the four scenarios is presented in the following result figures. The received bit rate of a single peer is always one of the four layers' cumulative bit rate presented in Table I, or 0 if no piece is received in time. In the result figures each value represents the average received bit rate for all 100 peers. Additional result details for each scenario are presented in Table III. Again, the results in the table represent the average values for all 100 peers.

The results in Figure 4 show the behavior of the algorithms under optimal bandwidth conditions. For this scenario only the results for the first minutes of the test run are shown, as all of the tested algorithms are able to constantly stream the best quality after the first few minutes under the given network conditions. During the start-up, the KP and the Zigzag algorithms both start at a higher quality and quickly increase

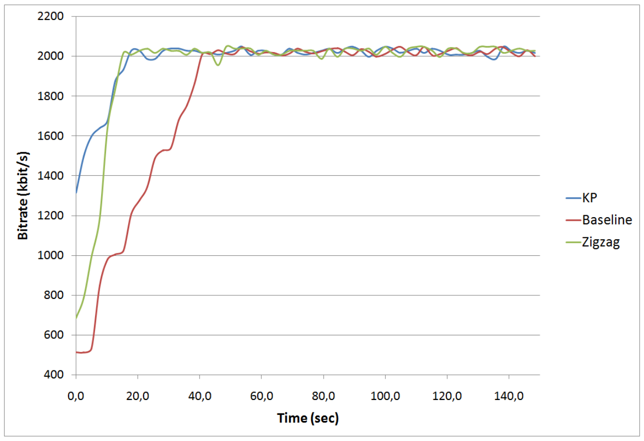

Fig. 4: Evaluation Scenario 1

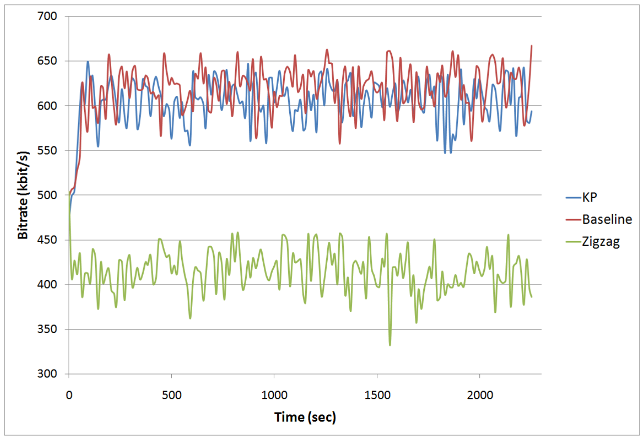

Fig. 5: Evaluation Scenario 2

the playback quality to the highest layer compared to the baseline algorithm. The reason for this behavior is that the KP and Zigzag algorithm assign a higher priority to pieces of higher layers with a close deadline than to lower layer pieces with a later deadline. The baseline algorithm, on the other hand, firstly fills the entire sliding window with lower layer pieces before switching to the highest quality, which leads to a lower playback quality in the beginning.

After the initialization, all three algorithms stream constantly the highest quality. All three piece-picking algorithms work as expected for Scenario 1, although the baseline algorithm takes unnecessarily long to reach the desired streaming quality.

In Figure 5 the behavior of the algorithms in the critical scenario is shown. The average bit rate varies significantly over the time of the streaming session for all algorithms since the bandwidth conditions lead to rather frequent quality switches. However, the quality switches occur at different time slots for the 100 peers, which leads to the steady variation in the average received bit rate.

In this scenario, the KP algorithm switches faster to the best quality possible under the network conditions than the baseline algorithm. However, once the baseline and the KP algorithm reach the average playback quality, the average quality of the

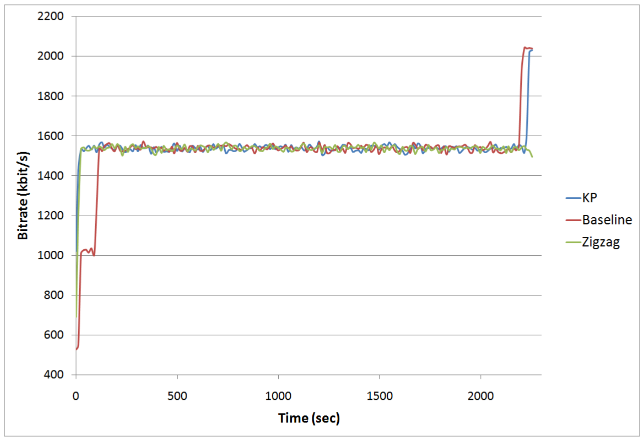

Fig. 6: Evaluation Scenario 3

KP algorithm is approx. 2% lower than the quality provided by the baseline algorithm. The reason for this difference is that the KP algorithm does not switch immediately to the higher quality once the first enhancement layer piece is received. When switching to the first enhancement layer, the baseline algorithm can usually only sustain the quality for a single time slot, and then needs to switch back to the base layer quality. This is due to the fact that the baseline algorithm cannot buffer the first enhancement layer because base layer pieces always have priority and the bandwidth is not sufficient to buffer both layers at the same time. The KP algorithm switches less frequently to higher quality; as it only switches to the higher quality once some enhancement layer pieces have been buffered, the KP algorithm can sustain the quality for more than three time slots on average.

The zigzag algorithm does not work well when the network conditions are critical. The clearly lower average streaming rate is due to the fact that the zigzag algorithms has to frequently skip time slots as the base layer pieces are not sufficiently prioritized for the given network conditions and no piece is received in time for playback. The skipping leads to a clearly lower average streaming rate and also to frequent disturbances of the user experience (as no frames are shown for the skipped time slots). Due to the skipping, the average streaming rate of all 100 peers is even lower than the base layer bit rate.

The results in Figure 6 show the behavior of the algorithms when more than sufficient bandwidth for the constant playback of three layers is provided. The average bit rate varies less for this scenario, as the bandwidth conditions allow the steady download of a consistent quality.

In this scenario, the zigzag algorithm switches most quickly to the desired playback quality, as the KP algorithm needs to buffer some enhancement layer pieces before increasing the quality. The baseline algorithm again has to fill the buffer for all lower layers first and thus switches later to the desired playback quality. After the initialization, all algorithms work similarly and provide a constant high playback quality. At the end of the streaming process, the baseline algorithm can

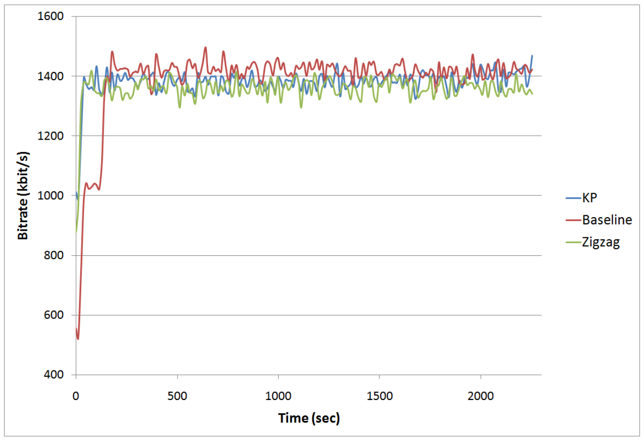

Fig. 7: Evaluation Scenario 4

TABLE III: Test Results (Average for 100 Peers)

Algorithm	**Bit Rate** [kbit/s]	**# Skipped Time Slots**	**# Quality Changes**
Scenario 1			
KP	1972	0	6.8
Baseline	1960	0	8.9
Zigzag	1970	0	9.7
Scenario 2			
KP	589	0.28	76
Baseline	603	0.19	346
Zigzag	406	108	287
Scenario 3			
KP	1501	0.07	8.4
Baseline	1484	0.05	10.1
Zigzag	1496	0.06	12.3
Scenario 4			
KP	1348	1.22	41.7
Baseline	1357	1.09	53.1
Zigzag	1327	4.90	79.1

increase the playback quality to the highest layer for approx. 45 seconds, as the buffers for all lower layers until the end of the stream have already been filled. The KP algorithm can also increase the quality at the end for approx. 20 seconds.

The final scenario illustrates the behavior of the algorithms when sufficient bandwidth for the download of three layers is provided and churn occurs. In this scenario, the behavior of the KP and the zigzag algorithms is similar. Both quickly go up to reach the highest sustainable playback quality and keep this quality until the end of the streaming session. However, the KP algorithm again performs fewer switches to a higher quality but can keep this quality longer. In average the KP algorithm can stream a slightly higher quality than the zigzag algorithm. It should be noted that this differences is due to more frame skipping events when applying the zigzag algorithm.

The behavior of the baseline algorithm differs for this scenario from the other algorithms. The baseline algorithm takes about 2 minutes to reach the sustainable quality, as the buffer filling for the lower layers takes longer. However, once it reaches the higher layers the average streaming rate is slightly higher than the one provided by the KP algorithm. As both, the KP and the baseline algorithm, provide a good performance, it depends on the usage scenario which algorithm is preferred (faster reaching of the sustainable quality or slightly higher average streaming rate).

V. Conclusion

In this paper, a piece-picking algorithm for layered content in Bittorrent-based P2P systems has been presented and compared to other piece-picking algorithms. To compare the algorithms, they have been implemented in the NextShare P2P system and real system tests have been performed in our test lab.

The results show that a piece-picking algorithm for layered content needs to consider the network conditions to find the best trade-off between ensuring continuous playback and providing the best possible quality. While the baseline algorithm performs well under low bandwidth conditions, it wastes time while going up to a better quality when the network conditions are good. On the other hand, the zigzag algorithm performs well under high bandwidth conditions, but when the network conditions become worse, unnecessary frame skipping occurs when applying the zigzag algorithm and the user experience is disturbed. The KP algorithm works well in high and low bandwidth situations, as it changes its parameters (i.e., urgency weighting, distortion reduction importance of the layers) based on the network conditions. This ensures that the best trade-off between ensuring continuous playback and providing the best possible quality for the network conditions is usually found. Additionally, the KP algorithm limits the quality switches to avoid disturbing the user experience.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant agreement no. 216217 (P2P-Next).

References

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[2] M. Zink, O. Knzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," in *International Workshop on Quality of Service*, 2003, pp. 137–154.

[3] N. Capovilla, M. Eberhard, S. Mignanti, R. Petrocco, and J. Vehkapera, "An architecture for distributing scalable content over peer-to-peer networks," in *MMEDIA Conference Proceedings*, June 2010, pp. 1–6.

[4] P2P-Next Project. http://p2p-next.org/.

[5] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "LayerP2P: Using layered video chunks in P2P live streaming," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1340–1352, November 2009.

[6] Y. Ding, J. Liu, D. Wang, and H. Jiang, "Peer-to-peer video-on-demand with scalable video coding," *Computer Communications*, vol. 33, no. 14, pp. 1589–1597, September 2010.

[7] R. Rejaie and A. Ortega, "PALS: peer-to-peer adaptive layered streaming," in *Proceedings NOSSDAV'03*. New York, NY, USA: ACM, 2003, pp. 153–161.

[8] M. Eberhard, T. Szkaliczki, H. Hellwagner, L. Szobonya, and C. Timmerer, "Knapsack problem-based piece-picking algorithms for layered content in peer-to-peer networks," in *AVSTP2P'10 Workshop Proceedings*, October 2010, pp. 71–76.

[9] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[10] Netem network emulation. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.