Adaptive Video Streaming: a Survey and Case Study

HU, Pili
MobiTeC, IE Department, CUHK
hupili@ie.cuhk.edu.hk
http://personal.ie.cuhk.edu.hk/~hpl011/

December 24, 2011

Abstract

In the past decade, Internet traffic has seen a significant change from web browsing to video viewing. The ongoing trend raises a challenging problem: how to stream data to heterogeous peers?

The designer of such data streaming architecture should bear the following considerations in mind: QoE, server load, network resource efficiency, scalability, etc. The heterogeneous peer network condition makes the design more complicated. The underlying codec ranges from Multi Description Coding to Multi Layer Coding. The data deliver architecture ranges from unicast, multicast, to P2P network. Researchers have focused on different system settings and optimization objectives.

This paper will first sum up several works in the context of adaptive video streaming. At the same time, we do a case study on a commercial adaptive video streaming system, which combines Multilayer Codec and P2P technology. Possible improvements on this system are proposed with reasoning. Some of the conjectures are verified through a corresponding simulation platform based on NS2.

Contents

1 Introduction

- User perceived experience.
- Vendor cost.
- User cost.
- System cost.

2 General Model

3 Problem Scope

3.1 Codec

3.2 Networking

4 Design of Adaptive P2P VoD System

4.1 Codec Choice

4.2 Transimission Protocol Choice

4.3 Overlay Construction

4.3.1 Construction with Prior Knowledge

4.4 Peer Selection

4.5 Buffer Management

4.6 Chunck Selection

MMKP, Knapsack
MCMF, Network Flow

4.7 Playback Decision

4.8 User Model

5 A Case Study and Simulation

Our simulation platform roots from a commercial deployment of P2P system[40], which has upgraded from P2P file downloading system, to P2P live streaming sytem, and then to the current version of multilayer P2P VoD system. Interested readers are recommended to [13] for further information. This platform has been validated against real trace data collected from 2008 Olympic period. It is widely used in P2P related research and pre-deployment testing.

Course IERG5270[41] has adopted this platform as a simulation foundation for several years.

5.1 System Description

To get a concrete view of the situation, we draw the version tree in fig(1). Blue blocks represent the real deployment of ASTRI. White blocks represent corresponding simulation platform. Red dash stands for strong equivalency. Green block shows the position of this project. The trial version of multilayer P2P VoD is forked from the original downloader platform and is developed by Zheng Wen in 2010.

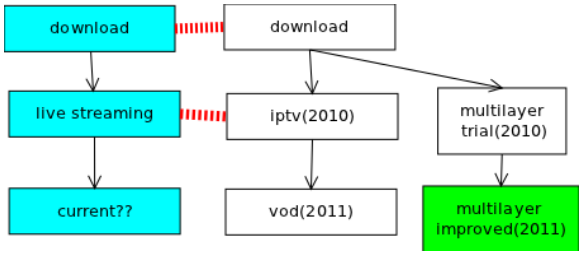


Figure 1: Version Tree of Real and Simulation

Precaution: the results obtained in this project may not reflect real system in every aspect(there is no red dash showing this kind of relationship). Nevertheless, methodology wise speaking, those results do provide some insights in Multilayer P2P system design.

5.2 System Benchmark Test

Before we start, system benchmark test of simulator performance is conducted.

Fig(2) shows how simulator running time varies with different number of cores. Note that PDNS [43] is the distributed version of NS[42]. In this paper, we just run the simulation on one machine with 16 cores. The original configuration of the platform involves 8 cores per simulation instance. Leaving some system recourses for other purpose, we can only run one simulation at a time, which significantly lowers our experiment efficiency.

We find that, when simulation time(measured in NS seconds) is small, 4 cores appear to be more efficient than 8 cores. Thus the following experiments are all issued to 4 cores to enhance our distributing level.

Fig(3) shows how simulator running time varies with the number of nodes simulated. It appears to be linear. The fit parameters are given in the title of the plot.

Fig(4) shows how simulator running time varies with the simulation time(in NS seconds).

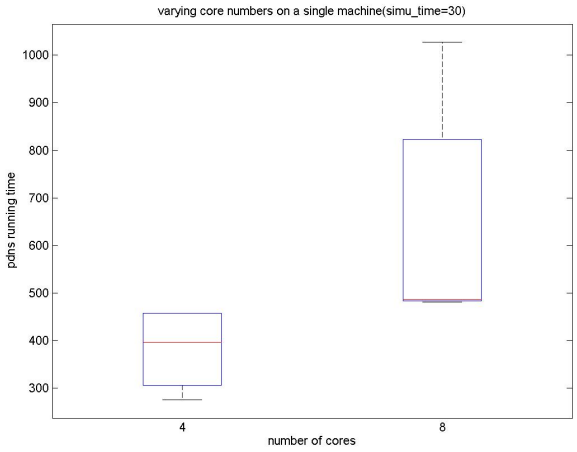


Figure 2: Running Time v.s. Number of Cores

With those system test, the final parameters we choose are:

- Number of Nodes: 160
- Simulation Time: 300 (NS seconds)

It's obviours that the number of nodes and simulation time is much smaller than those in a real VoD system. With those two configurations, the simulator can finish in about 2 hours using 4 cores. We sacrifice some equivalency to the real system to make the simulation tractable.

5.3 Experiment Configuration

In this study, we follow the same configuration given by Zheng Wen. The topology is summarized as follows:

- Star like. Peers are equally divided into 4 subnets. Every node connects to a single central router.
- Subnet parameters:
  - Subnet1: down:10Mb; up:10Mb.
  - Subnet1: down:3Mb; up:0.5Mb.
  - Subnet1: down:3Mb; up:0.5Mb.
  - Subnet1: down:3Mb; up:0.5Mb.

The layer configuration is summarized as follows:

- 3 Layers.

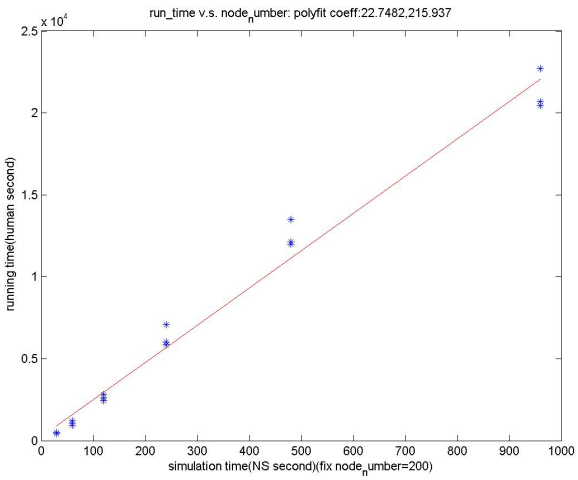


Figure 3: Running Time v.s. Number of Nodes

- Layer parameters:
  - Layer1: 256Kbit. (per piece)
  - Layer2: 256Kbit. (per piece)
  - Layer3: 512Kbit. (per piece)

Note that the cumulative data amount is 256Kbit, 512Kbit, and 1024Kbit. This is consistent with Wang's QoE study[30]. Thus we can rely on their QoE model to measure our system.

5.4 QoE Model Implementation

In Wang's paper[30], they seek for the tradeoff between bitrate(influenced by current layers subscribed) and discontinuity. They first do large amount subjective tests, and then fit the data into a continuous version formula, as is given in eqn(1).

$$MOS = c_1 \times d + \alpha \times (1 - e^{-b \times \lambda}) + c_2 \tag{1}$$

The parameters are $c_1 = -5$, $c_2 = 2$, $\alpha = 4$, $\lambda = 0.0015$.

In that paper, the maximum discontinuity considered is 50%, so we plot the MOS curve again to give the readers a better impression of how it looks like. See fig(5).

Fig(5(a)) is the 3D version and Fig(5(b)) is the contour version. We want to point out that by definition, MOS should be a score between 0 and 5. However, using this formula, we'll get negative scores sometime. Especially when the discontinuity is 100% and bitrate is 0, the MOS can be as low as

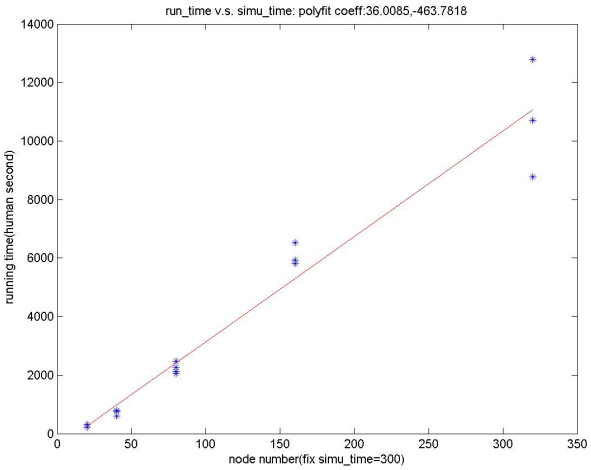
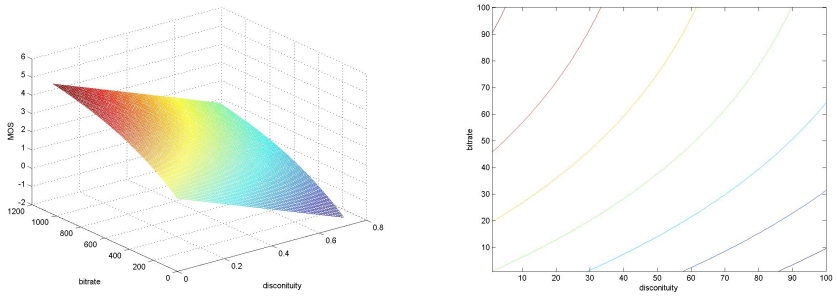


Figure 4: Running Time v.s. Simulation Time(NS)



(a) 3D (b) contour

Figure 5: Conclusion, QoE and Performance

-3. We don't regard this case as a bug. On the contrary, it is reasonable to assign negative opinions. In our implementation of this QoE model, peers report their statistics every 20 seconds. If the -3 score happens, that means the user is purely waiting in the whole 20 seconds, which causes a negative opinion naturally. Our system should work towards reducing this kind of disturbance.

5.5 Baseline Test

In this section, we evaluate the baseline using the QoE records reported by peers. We compute MOS using eqn(1), and take the average.

Fig(6) shows the result when total simulation time varies from 30 NS seconds to 960 NS seconds. It's interesting that with the increase of simulation time, the QoE gets improved, and the boxplot shows the statistics of

5 simulations with the same configuration. This means the improvement is statistically significant.

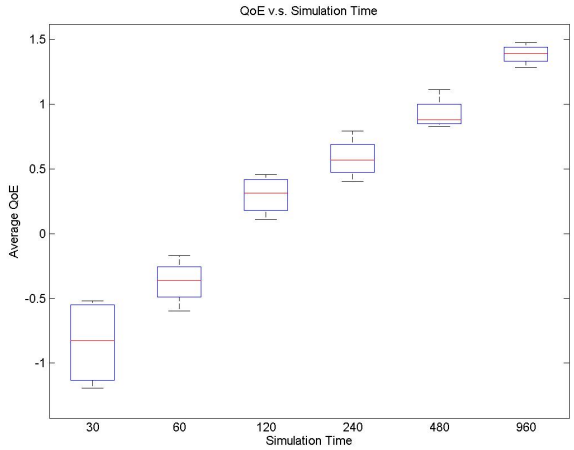


Figure 6: QoE v.s. Simulation Time

By examing the detailed trace file, we find that there are many negative opinions at the beginning of those simulation. Later after about 100 NS seconds, the reported MOS gets better and better. It makes sense that the system needs some bootstraping time, when all of the peers lack chunks. They can not help each other efficiently during this time.

We eliminate those negative records and make the boxplot again. See Fig(7). This figure verifies our conjecture. The last two boxes stand for 480s and 960s simulation time. There is little difference in mean within this range. Longer simulation time just makes the variance smaller. To explain, after some bootstraping period, the system approaches steady state and the average opinions become nearly a constant. The first box seems an outlier. This is because peers are joining the system gradually at that time. The number of reported records are very small. Only those peers connected directly to the source can get something to playback. Others all get negative opinions. By eliminating the negative opinions, the resultant score is very high.

Next, we pick a sample run out of all these baseline simulations. The global parameters and statistics are: simulation time is 480s; number of nodes is 160; number of QoE reports is 3749; number of nonnegative QoE reports is 2601; average QoE socre is 0.87; the nonnegative version of average QoE score is 2.02.

Fig(8) shows how QoE varies with curret time section. We can see the per section averaged QoE gets higher and higher in the first 150 seconds. After that, the section averaged QoE becomes steady. This again proves our

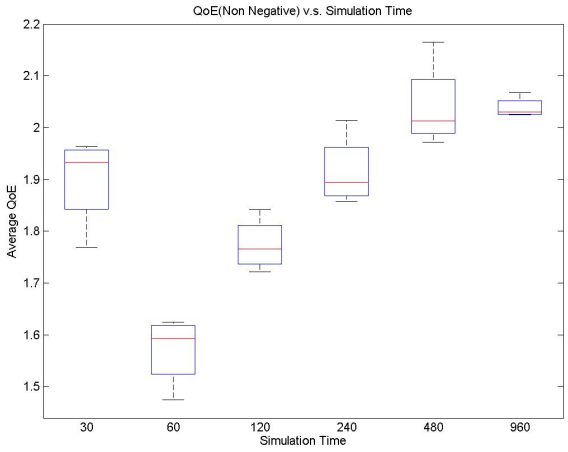


Figure 7: QoE(nonneg) v.s. Simulation Time

bootstraping conjecture.

Fig(9) shows the historgram of QoE reports from this sample run. We can conclude that, some powerful peers can get full 5 score experiene, whereas large amount of peers report negative scores. Our future improvements should mitigate this part(purely buffer for 20 seconds and play nothing).

5.6 Chunk Selection Architecture Reconstruction

In this section, we modify the original architecture to make it more clear and unified. This helps us develop strategies on this architecture later.

Original design separates chunk selection and peer selection. It is shown in Fig(10).

In this design, every peer is considered in a round robin fashion. After one neighbour is chosen, this peer selects chunks according to a three section based strategy. After one section is filled up to a certain ratio, the peer will consider the next section. Chunks in the same section are selected randomly. In our multilayer situation, layers are considered from lower ones to higher ones. That is, after the peer get 50% of the 3rd section, it will start to to fill the next higher layer. In order to reduce the network burden, peers are constrained to fetch chunks from no more than 20 neighbours simultaneously.

The improved architecture is illustrated in Fig(11). To make our decision more rationale and make the architecture clear, we collect all information first, and do peer selection and chunk selection together.

In this version, we adopt a very simple strategy, as is used in PALS [26]. This strategy makes decision on a chunk basis. It first consider base layer chunks and then moves to enhancement layers. In one layer, the chunks

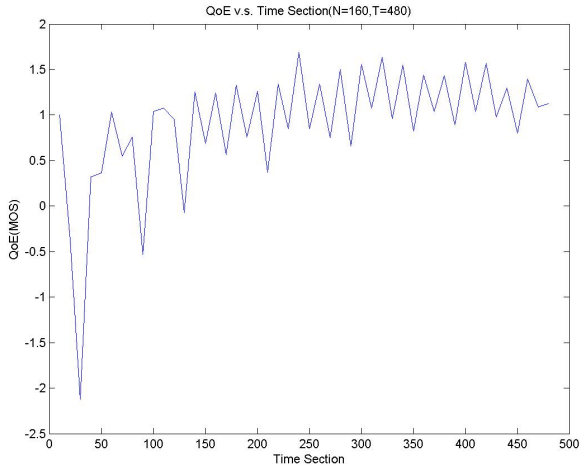


Figure 8: QoE Varies with Current System Time

closer to playback points are considered first. The overall picture is like a snake-shaped move on the 2 dimension buffer map. Fig(12) illustrates this strategy. If there are more than one peers available for one chunk, we just randomly choose one peer.

After each modification, we evaluate the result from two aspects:

- QoE. The formula is given in eqn(1). In our study, we didn't make any aggressive strategies like scheduling all requests to the server. So we don't subtract server's contribution from this metric intentionally. Another reason is, in the underlying overlay construction(which is not modified in this study), the server is treated as a normal peer. Not all peers can turn to server for help when the lack chunks.

- Performance. Another important thing we care is the performance of simulator. This kind of packet level simulation is very time consuming. One iteration of strategies calls for several hours of running typically. In order to fulfil a short-term project, we can not make strategies of high time complexity. What's more, the running time of simulator reflects the computation resource consumption in real deployments. User experience will degrade if their player consumes extra large amount of resources, to limit other process.

The result of modification in this section is:

- QoE: $0.7 \rightarrow 3.3$

- Performance: nearly doubled.

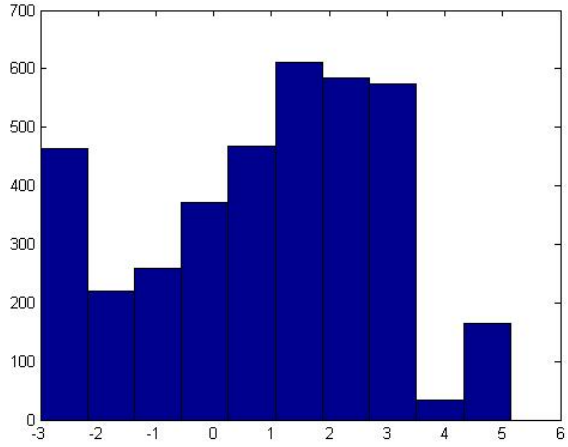


Figure 9: Histogram of QoE Distribution

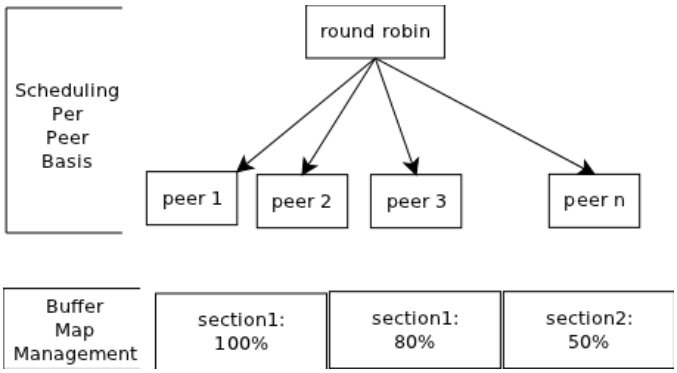


Figure 10: Original Architecture

5.7 Priority Based Upgrade

In this section, we do more engineering upgrades. We abstract the chunk selection strategy using a priority table. One sample table with window size equal to 5 is given in Table(1). This table is equivalent to PALS.

Further investigation on optimal priority table can be done easily in this framework. Result of this modification is:

- QoE: nearly the same.
- Performance: decrease slightly.

5.8 Scalable Window Size

After previous upgrades, we examine the trace files from several simulations. From the trace, we find many powerful peeers can get full 3 layers in the

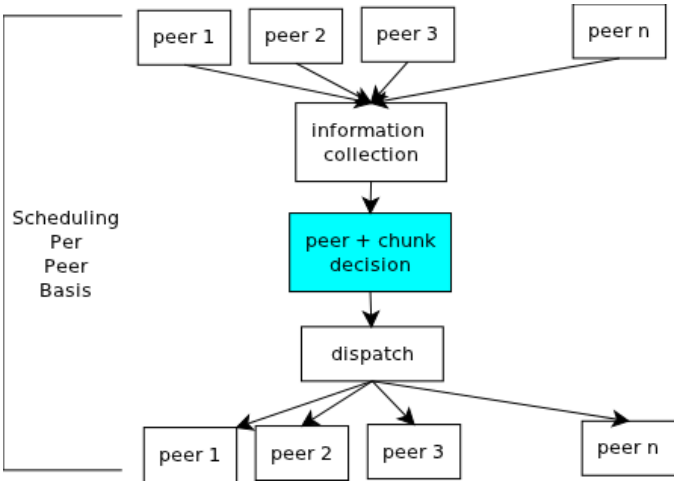


Figure 11: Improved Architecture

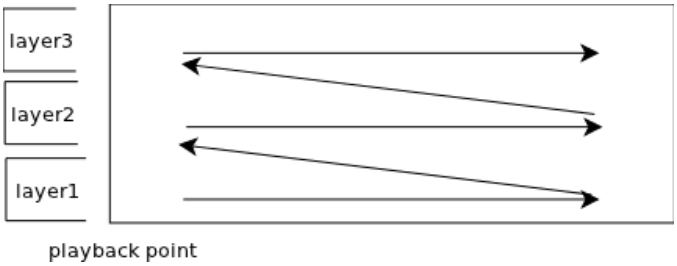


Figure 12: PALS Style Chunk Selection

whole window(60 seconds, 3 layers). The restriction on window size is a special desgin of video streaming, no matter live or VoD. This is because chunks very far from playback point is not of high interest to the peer. Every peer only needs to store enough chunks in the buffer to guarantee smooth playback. However, there is a side effect. Capable peers who are playing the early part can not help others who are playing the later part. Thus we want to give those peers a chance to help others. This is the rationale behind scalable window size.

Table(2) illustrates a scaled window with the size doubled. Using this table, less capable peers will act as they are using table(1). For those pow-

Table 1: A Sample Priority Table with Window Size = 5

	t1	t2	t3	t4	t5
layer3	11	12	13	14	15
layer2	6	7	8	9	10
layer1	1	2	3	4	5

Table 2: A Sample Priority Table with Window Size = 5+5

	t1	t2	t3	t4	t5	t6-t10
layer3	11	12	13	14	15	...
layer2	6	7	8	9	10	...
layer1	1	2	3	4	5	...

erful peers, after they finish the first section, as is in table(2), they can move to the second section. This is the simplest illustration of the idea of scalable window size. Ideally, the window size should be computed on recipient of heartbeat message. When a peer updates the network connection measurements, it can compute a proper window size based on those data. Last but not least, the priority table should be recomputed if the window size changes.

In this version, we fill the second section of the window using the same strategy as used for the first section. The result is:

- QoE is improved a little.
- Performance degrades sharply.

5.9 Performance Optimization

After last experiment, one simulation needs about 4.5 hours to complete. This is really time consuming. In order to do more iterations, we try to optimize the simulator performance right away. At first, optimization described in this section only aims at the performance of simulator. However, it turns out the QoE is also improved. We'll talk about this phenomenon in conclusion section.

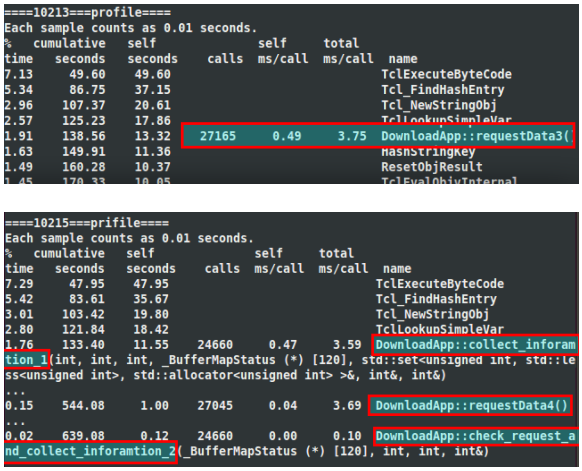


Figure 13: GNU Profile, Output

Fig(13) shows two screenshots of the output of 'gprof'. We first locate the most time consuming part that we can control. It turns out to be the 'requestData3()' function, which implements the strategy mentioned in last section. We then separate some sub functions from this one and find that the information collection is most time consuming. Note that we do not send probes to collect information. Instead, peer information is updated on recipient of periodical notification message. The information collection sub function only iterates over some map structures and fill the availability information into one unified 2D buffer map.

According to the analysis of the profile, we optimized those iteration process. For more details, please refer to the code repository of this project. Difference can be found between the following two commitments:

`b45038404d8970c8ee1654d9bdbb10703432b8cd`
`5d3aabdb1546d88da6e7a026d2d1fe67b04ad2f1`

Result of the optimization is:

- QoE: improved a little.
- Performance: 4.5h $\rightarrow$ 3.5h.

Note that this is not the end of the performance optimization. One of the possible directions is to optimize the sampling method using a reservoir[29]. However, this optimization is conducted with the knowledge that we'll only random select one peer if there are multiple peers available for one chunk. This kind of optimization is too aggressive and will limit the design space of strategies, so we don't do them in this research version. Designers can make notes and do this kind of optimization when all strategies are fixed in a real release version.

5.10 Introduce Randomness in Second Window Section

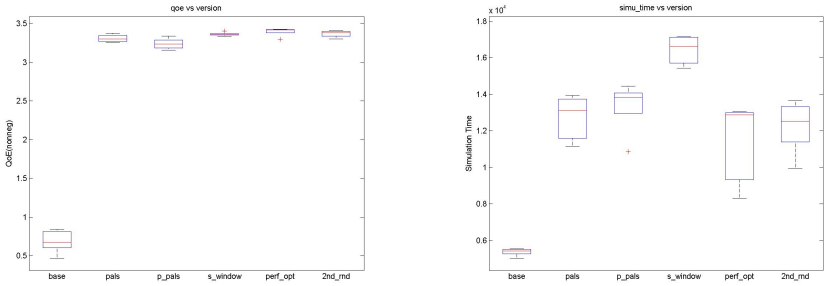
Due to time issue, our last trial based on the already established mechanism is to introduce randomness in second window section. As is known, randomness is a good property in the sense that it helps the system to spread chunks uniformly. The first window section is scheduled using a snake-shaped priority, this design considers user-perceived experience. However, the second window section is for those powerful peers. Those chunks are not hurry for playback. Thus we make the second section of the priority table randomly initialized. Every peer will have different priority table.

The result is:

- QoE: get worse.
- Performance: get worse.

5.11 Conclusion of the Case Study

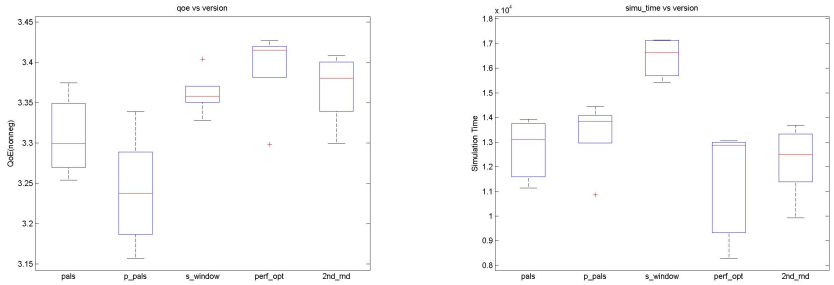
In this section we put together the result of 6 big versions. See Fig(14). In terms of QoE, version 2 improves a lot from baseline. After that, there are not so much difference in QoE. In terms of simulator performance, all modified versions are slower. The slowest version is the scalable window one. From the plot, we know our performance optimization really helps to reduce running time.



(a) qoe (b) time

Figure 14: Conclusion, QoE and Performance

Fig(15) gives a closer look at the 5 modified versions. The most interesting observation is the difference before and after performance optimization. Note that in section(5.9), we did nothing about strategy. It is simply equivalent code reconstruction. However, while we observe better performace of the simulator, we also find the QoE is improved a little. From the box plot, this improvement seems statistically significant(there are two ourliers).



(a) qoe (b) time

Figure 15: Conclusion, QoE and Performance

Similar phenomena have been observed in our other VoD simulations using the same series ASTRI platform[13]. When the server is idle, the results tends to be better. Just like in our experiment, after performance

optimization, the computation burden is lowered, thus we see better results. One conjecture is that this phenomenon is due to underlying timing facilities, either NS[42, 43] layer or the p2p simulator layer[13]. The verification of this point is left as future work.

6 Conclusion

Here we conclude two pieces of experience from this study:

- Engineering approach v.s. academic approach: where is the biggest cake? From section(5.11), we see the most significant improvement lies in the first engineering upgrade. Although the modification itself reads simple, it makes the strategy framework more clear. Upon this framework, one simplest PALS like strategy achieves very good result. By examining the simulator carefully, we believe more engineering optimization can be found. Without an ideal underlying platform, the verification of algorithms on it is questionable. At same time, the benefit we tried exhaustively to get using better strategies may come easier with some engineering upgrades.
- Time distribution of this project:
  - 70%, literature survey.(30+ papers)
  - 15%, environment setup, bugfix of the platform.
  - 5%, first unified version(QoE:0.7→3.3, biggest improvement in this study)
  - 10%, scalable window, performance optimization, random 2nd section. (little outcome)

  Sometimes we try very hard, to get little outcome. Sometimes, ideas flash through our minds, and suddenly we get significant achievements.

7 Future Works

Here we list some potential research topics in an adaptive video streaming system:

-

For multilayer P2P VoD system, we list some potential research topics:

- Degree composition and decomposition.
- If we use the framework given in section(5.7), what's the optimal priority table. In order to achieve QoE aware design, models like [30] should be considered. Due to time limit, we haven't deducted the optimal table in this study.

- As for the simulator, how to explain the relationship between QoE and simulator performance. Theoretically speaking, QoE should only depend on strategies and irrelevant of simulator performance.

Acknowledgements

The author would like to thank course instructor of IERG5270, Professor DM Chiu, and course TA Tom Fu. The inspiring discussion with professor Wing Lau is very helpful. That helps the author find related works in the context of IP multicast. The support from ASTRI[40] is also important, which helps us get a better understanding of the commercial deployment of a multilayer P2P VoD system. The baseline platform is developed by Zheng Wen, HKU. Special thanks to those coding.

Appendix

References

[1] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee. On multiple description streaming with content delivery networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1736–1745. IEEE, 2002.

[2] J.G. Apostolopoulos. Reliable video communication over lossy packet networks using multiple state encoding and path diversity. In *Visual Communications and Image Processing*, volume 1. Citeseer, 2001.

[3] S. Bhattacharyya, J.F. Kurose, D. Towlsey, and R. Nagarajan. Efficient rate-controlled bulk data transfer using multiple multicast groups. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1172–1179. IEEE, 1998.

[4] J. Byers, M. Luby, and M. Mitzenmacher. Fine-grained layered multicast. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 1143–1151. IEEE, 2001.

[5] Y. Cui and K. Nahrstedt. Layered peer-to-peer streaming. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 162–171. ACM, 2003.

[6] L. Dai, Y. Cui, and Y. Xue. Maximizing throughput in layered peer-to-peer streaming. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 1734–1739. IEEE, 2007.

[7] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*, pages 362–373. ACM, 2011.

[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 *Proceedings of the 2010 ACM workshop on Advanced video streaming techniques for peer-to-peer networks and social networking*, pages 71–76. ACM, 2010.

[9] T.Z.J. Fu, D.M. Chiu, and Z. Lei. Designing qoe experiments to evaluate peer-to-peer streaming applications. In *Visual Communications and Image Processing*, 2010.

[10] T.Z.J. Fu, W. Leung, P. Lam, D.M. Chiu, and Z. Lei. Perceptual quality assessment of p2p assisted streaming video for chunk-level playback controller design. In *Packet Video Workshop (PV), 2010 18th International*, pages 102–109. IEEE, 2010.

[11] A.E.L. Gamal and T. Cover. Achievable rates for multiple descriptions. *Information Theory, IEEE Transactions on*, 28(6):851–857, 1982.

[12] V.K. Goyal. Multiple description coding: Compression meets the network. *Signal Processing Magazine, IEEE*, 18(5):74–93, 2001.

[13] J. Huang, G. Cheng, J. Liu, D.M. Chiu, K. Wu, and Z. Lei. A simulation tool for the design and provisioning of p2p assisted content distribution platforms. under review?, 2010.

[14] S.K. Kasera, G. Hjalmtusson, D.F. Towsley, and J.F. Kurose. Scalable reliable multicast using multiple multicast channels. *Networking, IEEE/ACM Transactions on*, 8(3):294–310, 2000.

[15] S. Khan, K.F. Li, E.G. Manning, and M.D.M. Akbar. Solving the knapsack problem for adaptive multimedia systems. *Stud. Inform. Univ.*, 2(1):157–178, 2002.

[16] J. Liu, B. Li, and Y.Q. Zhang. Adaptive video multicast over the internet. *Multimedia, IEEE*, 10(1):22–33, 2003.

[17] Z. Liu, Y. Shen, S.S. Panwar, K.W. Ross, and Y. Wang. P2p video live streaming with mdc: Providing incentives for redistribution. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 48–51. IEEE, 2007.

[18] Z. Liu, Y. Shen, S.S. Panwar, K.W. Ross, and Y. Wang. Using layered video to provide incentives in p2p live streaming. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pages 311–316. ACM, 2007.

[19] Z. Liu, Y. Shen, K.W. Ross, S.S. Panwar, and Y. Wang. Layerp2p: using layered video chunks in p2p live streaming. *Multimedia, IEEE Transactions on*, 11(7):1340–1352, 2009.

[20] N. Magharei and R. Rejaie. Adaptive receiver-driven streaming from multiple senders. *Multimedia Systems*, 11(6):550–567, 2006.

[21] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 117–130. ACM, 1996.

[22] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 177–186. ACM, 2002.

[23] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. *Peer-to-peer systems IV*, v:127–140, 2005.

[24] M. Qin and R. Zimmermann. Improving mobile ad-hoc streaming performance through adaptive layer selection with scalable video coding. In *Proceedings of the 15th international conference on Multimedia*, pages 717–726. ACM, 2007.

[25] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for congestion controlled video playback over the internet. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 189–200. ACM, 1999.

[26] R. Rejaie and A. Ortega. Pals: peer-to-peer adaptive layered streaming. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 153–161. ACM, 2003.

[27] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9):1103–1120, 2007.

[28] T. Szkaliczki, M. Eberhard, H. Hellwagner, and L. Szobonya. Piece selection algorithm for layered video streaming in p2p networks. *Electronic Notes in Discrete Mathematics*, 36:1265–1272, 2010.

[29] J.S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[30] Jingjing Wang, Z. J. Tom Fu, Dah Ming Chiu, and Zhibin Lei. Perceptual quality assessment on b-d tradeoff of p2p assisted layered video streaming. In *VCIP*, 2011.

[31] Y. Wang, A.R. Reibman, and S. Lin. Multiple description coding for video delivery. *Proceedings of the IEEE*, 93(1):57–70, 2005.

[32] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[33] M. Wien, H. Schwarz, and T. Oelbaum. Performance analysis of svc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(9):1194–1203, 2007.

[34] X. Xiao, Y. Shi, and Y. Gao. On optimal scheduling for layered video streaming in heterogeneous peer-to-peer networks. In *Proceeding of the 16th ACM international conference on Multimedia*, pages 785–788. ACM, 2008.

[35] X. Xiao, Y. Shi, Y. Gao, and Q. Zhang. Layerp2p: A new data scheduling approach for layered streaming in heterogeneous networks. In *INFOCOM 2009, IEEE*, pages 603–611. IEEE, 2009.

[36] X. Xiao, Y. Shi, B. Zhang, and Y. Gao. Ocals: A novel overlay construction approach for layered streaming. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 1807–1812. IEEE, 2008.

[37] J. Zhao, F. Yang, Q. Zhang, and Z. Zhang. On improving the throughput of media delivery applications in heterogenous overlay network. In *Proc. IEEE Globecom*. Citeseer, 2006.

[38] Youtube, Demo of User Controlled Adaptation, `http://www.youtube.com/`

[39] Pili Hu, 2011, Lightweight Distributing Toolset, `https://github.com/hupili/Lightweight-Distributing-Toolset`

[40] ASTRI, `http://www.astri.org/`

[41] IERG5270, 2011, Books, Course Slides, Tutorial Slides, Classmates' Presentations, etc. Not publicly available. Please contact course instructor or TA for those materials. `http://course.ie.cuhk.edu.hk/~ierg5270/`

[42] Network Simulator 2, `http://isi.edu/nsnam/ns/`

[43] Parallel Distributed NS, `http://www.cc.gatech.edu/computing/compass/pdns/index.html`