Aggregate Congestion Control for Peer-to-Peer File Sharing Applications*

Wei LI[1], Shanzhi CHEN[1, 2], Yaning LIU[1], Xin LI[1]
([1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications; [2]China Academy of Telecommunication Technology)
Python.Gozap@gmail.com

Abstract

Peer-to-Peer file sharing applications, which enable peers to establish multiple TCP connections between other peers to transfer data, pose new challenge to congestion control. Since conventional congestion control only aims to make each of those connections TCP-friendly, self users can increase the number of connections to grab a large share of the bandwidth, introducing more congestion and degrading the overall network performance. To address this issue, in this paper we propose and design an aggregate congestion control mechanism called ACCM which enforces the friendliness of network upon all the connections belongs to an application instead of upon individual connection. By observing the share congestion in access link through application-level measurement technology, ACCM dynamically adjusts the window size of parallel TCP connections, achieving friendliness to the network on the basis of maximize utilization of network bandwidth. The simulation experiments demonstrate that certain fairness and congestion avoidance can be achieved in presence of congestion and the network bandwidth can be effectively utilized in absence of congestion with ACCM.

1. Introduction

In recent years, file applications based on Peer-to-Peer (P2P) paradigm have become more and more popular. The P2P approach differs from the traditional client/server approach for building network applications since the participating hosts play dual roles as servers and clients. Thus, a peer generates workload for the P2P application, while also providing the capacity to process the workload generated by other peers.

The novelty of the P2P paradigm relies on two main concepts: cooperation among users and resource sharing. The new concept according to which the users of P2P systems provide services for the community has several beneficial effects on the global system performances: it permits to increase the service capacity; it improves the network service reliability; it makes the network more flexible and adaptive to the users' wishes.

Nevertheless, until recently P2P-based applications (such as BitTorrent [1] and Kazaa [2]) have mainly been developed for the sharing of audio and video files. P2P-based file sharing applications are characterizing a great fraction of the Internet traffic nowadays and several statistics on IP traffic have recently put in evidence that P2P traffic is starting to dominate the bandwidth in certain segments of the Internet. In particular, a recent study by CacheLogic [3] shows that a high fraction of Internet traffic can be contributed to about 60-70% of the traffic in Internet and about 80% of the traffic in the last-mile networks.

P2P file sharing applications, which enable peers to establish multiple connections between other peers to transfer data, pose new challenge to congestion control. Since conventional congestion control only aims to make each of those connections TCP-friendly, self users can increase the number of connections to grab a large share of the bandwidth, introducing more congestion and degrading the overall network performance. It makes the ISPs' networks, especially the last-mile networks, become congested network bottlenecks. The impacts of P2P file sharing applications on network traffic patterns, capacity planning, and infrastructure upgrades is significant. P2P file sharing traffic causes network congestion and deteriorates the performance of Internet traditional applications, ultimately leading to ISPs customers

* This work was supported by the National Natural Science Foundation of China under Grant No.60672086; the National 863 Program of China under Grant No. 2006AA01Z229.

978-0-7695-3263-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SNPD.2008.98

dissatisfy and churn.

To address this issue caused by aggressive P2P file sharing applications, in this paper we propose *ACCM* which is an aggregate congestion control mechanism for P2P file sharing applications. The main goal of *ACCM* is ensure that the P2P file sharing applications does not negatively contribute to the share congestion of the access link during overload periods. By observing the share congestion in access link through application-level measurement technology, *ACCM* dynamically adjusts the window size of multiple parallel TCP connections, achieving friendliness to the network on the basis of maximize utilization of network bandwidth. When the access link is idle, *ACCM* increase the TCP connection window size, improving the efficiency of data transfer and the utilization of bandwidth; but when observing the share congestion of access link, *ACCM* will decrease the TCP connection window size, preventing those P2P file sharing applications contribute too much to the share congestion. Furthermore, *ACCM* requires neither network node support nor transport layer modification, which makes it easy to integrate into existing applications.

The remainder of this paper is organized as follows: Section 2 gives a short review of some prior works in aggressive congestion control. Section 3 describes the details of our *ACCM* mechanism. Experiments and analysis are given in Section 4. The conclusion is drawn in the last section.

2. Related work

Compared with congestion control in unicast, aggregate congestion control for multiple flows is relative new. The core idea of aggregate congestion control is how to guarantee the fairness of a group of flows.

Banchs [16] has recently proposed the notion of user fairness, in which every sender is considered as a single user in the network. This fairness scheme, referred to as user fair queuing (UFQ), suggests that all connections started by a user (sender) should be considered as a single entity for rate allocation.

UFQ approach needs ISPs deploy new equipment or upgrade existing equipments. Although control in the network side is very efficient, but it also will cost ISPs huge. Therefore, many researchers have proposed many user-side aggregate congestion control approach.

Congestion Manager (CM) [5] uses one AIMD congestion window adjustment loop for the flow aggregate to achieve a fair combined throughput. CP [6] adopts equation-based rate adaptation [7] with packets subsampling to achieve fair bandwidth share. MPAT [8] keeps multiple bandwidth estimation loops and allows the application to allocate bandwidth to different flows while ensuring that the total throughput is fair. Hacker et al. study parallel TCP flows [9] and mimic TCP flows with longer RTT, so that flows in the aggregate consume less bandwidth than a TCP flow, making the aggregate TCP-friendly. 4CP [10] adjusts its cwnd through balancing the loss probability along a network path to some target of loss probability with farsighted strategy. ImTCP-bg [11] uses an inline network measurement technique [12] which uses modified TCP protocol to measure the network available bandwidth. Based on measured available bandwidth, ImTCP-bg adjusts the rate of data packets transfer.

Compared to the network-side algorithms such as URQ [16], the above algorithms are all implemented in the user side, will not cost ISPs even a penny. The fundamental flaw is that they all adopted cross-layer design idea, need modify TCP protocol. Although experiment results show the effectiveness of those approach, but it is difficult to deployment. Therefore, a new congestion control method is therefore required

2. Aggregate congestion control mechanism

As illustrated in Figure 1, our *ACCM* mechanism consisting of the following two core algorithms: one is access link congestion inferring algorithm and the other is connection window control algorithm. Access link congestion inferring algorithm is based on application-layer measurement technology. We send probe packet to the paths belong to the connections of P2P file sharing application and measure the RTT and packet loss rate to detect the status of access link. Connection window control algorithm is similar to TCP window control; the difference is that we are control the TCP connection in the application layer. When congestion occurred in the access link, not only the underlying TCP transfer window will backoff, but also in the application layer the TCP connection window will decrease with AIAD strategy. Thus, P2P file sharing application could ensure the fairness of other single-connection applications such as FTP.

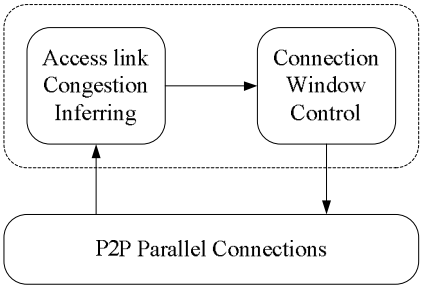

Fig.1. Framework of ACCM

Below, we will sense the access link congestion inferring algorithms and connection window control algorithm described in detail.

3.1 Access link congestion inferring

In most P2P file sharing application, transfer files are often split into fixed-size fragments (such as 256K in BitTorrent [1] or Gutella [2]), and all the fragments are distributed in different peers. When one peer named P want to retrieve a file, it will establish multiple TCP connections between many other different peers to transfer data as shown in Figure 2. These TCP connections all have the same source node (P) but the different destination node. When Congestion occurred in the network, if these connections belong to peer P all have shared Congestion link, then the link must be the access link of peer P, because they are connected to the same source node but the destination node is different , the only one link they shared is the access link of the source node. Similarly, if these connections belong to peer P are all have not shared congestion link, then the access link of the source node P must be in idle state. Therefore, from above analysis we know how to detect whether those link has difference destination node are all share congested link is the core idea of our access link congestion inferring algorithm.

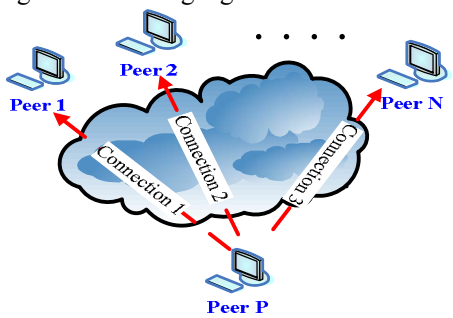

Fig.2. Parallel download of P2P file sharing application

Sharing congestion detection [12, 13, 14, 15] now is a hot research topic. The basic technique is based on the observation that measured delays of two paths show strong correlation if the paths share one or more congested links and little correlation if they do not share any congested links. Min Ski Kim proposed a new shared congestion detection approach named wavelet-based technique. Compared with other techniques, it provides faster convergence and higher accuracy while using fewer packets. Furthermore, the denoising process effectively removes noise and makes it more resilient to synchronization offset, which confuses other techniques. For details of this technique, please refer to the original paper [15].

Our access link congestion inferring algorithm is based on the Wavelet-based Share Congestion Detection technique proposed in [15]. Next, we will describe the algorithm in detail and give the pseudo-code in Table.1.

We use *XCOR(X, Y)* to denote the cross-correlation coeffient of one-way delay sequence of path *X* and *Y*. When *XCOR(X, Y)* returns 1, the two path share a congested link; when it returns 0, no shared congested link is detected. Suppose node *P* has established *N* connections with other peers for data transfer, and those connections has *M* difference destination. Our algorithm consists of two stages: sampling and processing. In the sampling stage, node *P* will first randomly select *M* connections that has difference destination from all *N* connections, and sends to every destination node a sequence of UDP packets (the number is *K*) with a timestamp, starting at time *T0* with its own clock. Each such UDP packet is called a probe packet. Probe packets are sent at a constant rate until *T0+T*, where T is the probe interval. On receiving a probe packet, every destination node calculates one-way delay and sends it, with the original timestamp, back to the source node *P*. Then node *P* records the one-way delay together with the timestamp as a delay sample. The sampling stage ends when the last delay sample from destination node is received (or upon timeout if the last probe or the reply is lost), and at the end node *P* will records the probe packet lost rate of each connections. In the processing stage, if on connection suffers probe packet loss, which means the access link is idle, and we do not need to do anything. When some connection such as connection j has suffered probe packet loss, we will compute the cross-correlation coefficients of connection j and all other connection and the aggregate result. If the aggregate result is equal to 0, it means the access link of node *P* is idle. If the aggregate result is equal to M, it means the access link of node *P* is congested. If other

scenarios, we think the access link of node *P* is in relative steady state, may be congestion was occurred in other link but not in the access link of node *P*.

Table.1. Pseudo-code of Access link congestion inferring algorithm

```
1.  select M connections that has difference
    destination from all N connections into C;
2.
3.  /* sampling stage */
4.  for (i = 1; i ++; i =< M) {
5.     send probe packet for C(i);
6.     measure RTT and packet loss rate of C(i);
7.  }
8.
9.  /* processing stage */
10. flag = 0;
11. if (isPacketLoss(C(j)) {
12.    / * means flag = sum_{i=0}^{M} XCOR(i,j) */
13.    for (i = 1; i++; i =< M) {
14.       flag += XCOR(i, j);
15. }
16.
17. if (flag == M)
18.    the access link of Peer P is congested;
19. else if (flag == 0)
20.    the access link of Peer P is idle;
21. else
22.    the access link of Peer P is stable;
23. }
```

3.2 Connection windows control

Each P2P file sharing application controls all of its data transfer connections with a single instance of *ACCM*. *ACCM* maintains a window of outstanding TCP connections: it will only issue a new data transfer connection when the network in the idle state. The pseudo-code is given in Table.2.

Table.2. Pseudo-code of Connection window control algorithm

```
1.  switch (the congestion status) {
2.     idle:
3.        W = W + ⊿;
4.     congested:
5.        W = W - ⊿;
6.     stable:
7.        W = W;
8.  }
```

ACCM maintains a current connection window size *W* in a manner similar to that of TCP window control. In our algorithm, we use AIAD strategy to dynamically adjust the connection window size. We set the initial window size as W_0 and the maximum as W_M. When *ACCM* show the access link is idle, it increases *W* by ⊿, improving the efficiency of data transfer and the utilization of bandwidth, but When the access link is congested, *ACCM* will decrease *W* by ⊿, preventing those P2P file sharing applications contribute too much to the share congestion and keeping fairness to other applications. In other scenarios, network is in a stable state, the window size *W* will remain unchanged.

4. Performance evaluation

In this section we will evaluate the performance of our aggregate congestion control mechanism for P2P file sharing applications. In our experiment, we choose BitTorrent [1] which is the most famous P2P file sharing application in Internet, and will implement *ACCM* in XBT [17] (an open source implementation of BitTorrent protocol, written in C++). We will do following performance evaluation in the experiment: fairness between BitTorrent and FTP, network bandwidth utilization and algorithm overhead.

4.1 Experiment setup

The topology and configuration of the testbed is illustrated in Figure 3. We have a testbed with 12 PCs interconnected by some Cisco routers and we create an experiment network with three sub networks. In order to run our experiment with a relatively large number of peers, we configure two IP addresses in each PC and bind each peer with a particular IP address, then we will have 24 peers. The uplink bandwidth of each peer is 500*Kbps* and downlink bandwidth is 1*Mbps*.

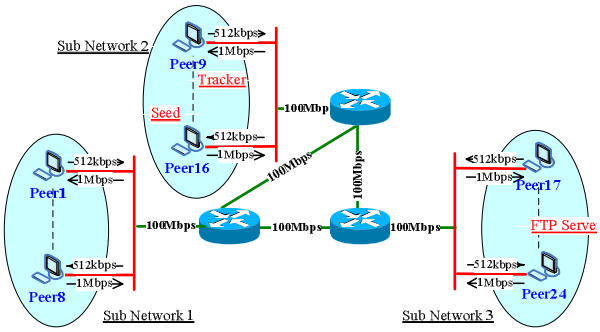

Fig.3. the topology of the testbed

We have implemented *ACCM* mechanism in XBT [17], and the paramtre setting of *ACCM* is shown in. We deploy the BitTorrent tracker server on *Peer9* and

the seed on *Peer10* in sub network 2, and deploy one ftp server on *Peer20* in sub network 3. The standand BitTorrent client and modified BitTorrent client are all deployed on each other peers, and the ftp client only deployed on *Peer1*.

We prepared two file all having a size more than 500 MB for BitTorrent downloading and FTP downloading (The size of file to be delivered is big enough, so one of our experiments can be sustained for a longer period of time) and we will measure the throughput of FTP application and BitTorrent application and access link bandwidth utilization in *Peer1*.

Table 3: ACCM PARAMETERS

The number of probe packets (K)	10
Probe interval (T)	1 (s)
Initial window size (W_0)	1
Max window size (W_M)	32

4.1 Fairness between BitTorrent and FTP

Figure 4 shows the aggregated throughput comparison result of FTP application and standard BitTorrent application. Before the time 200s, there is only FTP traffic in the network, and the throughput of FTP application are all very stable. When we start standard BitTorrent application in the network, it is aggressive to grab most of bottleneck bandwidth from FTP application. And as shown in Figure 4, the throughput of FTP application decreased over time.

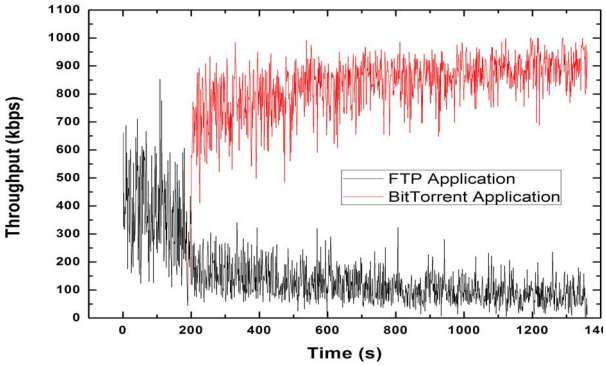

Fig.4. The aggregated throughput of FTP and Standard BitTorrent

Figure 5 shows the aggregated throughput comparison result of FTP application and BitTorrent application with *ACCM*. After we started the modified BitTorrent application (with *ACCM*), the throughput of FTP application has decreased, but still very stable. Its means *ACCM* could efficiently enable BitTorrent application keep fairness with FTP application.

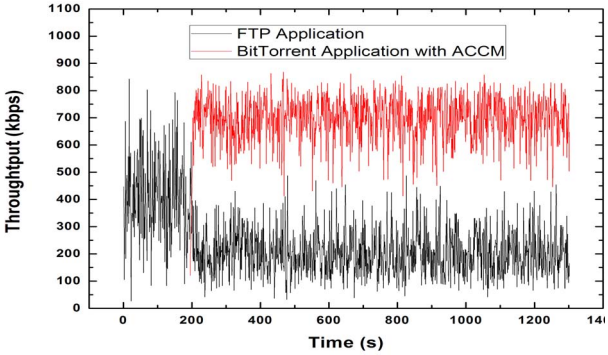

Fig.5. The aggregated throughput of FTP and BitTorrent with ACCM

4.2 Network utilization

We compare the network utilization between standard BitTorrent and BitTorrent with *ACCM* and the result is show in Figure 6. Before the time of 200s, there is only BitTorrent traffic in the network. Then we inject same number of FTP flows into the network. At the beginning of the experiment, the network utilization under Standard BitTorrent increase faster than under BitTorrent with *ACCM*, because *ACCM* enable BitTorrent use AIAD strategy to increase its connection window to alleviate the impact on Internet network and traditional Internet traffic. The comparative result from 30s to 200s shows BitTorrent with our *ACCM* mechanism also can achieve high network utilization as standard BitTorrent. Network utilization changes more drastically in scenarios of BitTorrent with *ACCM*, which shows the *ACCM* mechanism worked as soon as the access link was congested.

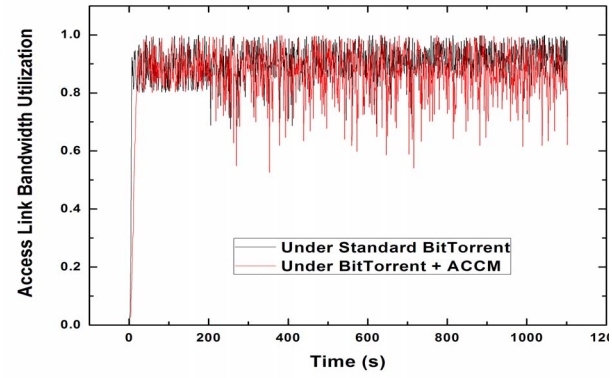

Fig.6. The access link utilization under Standard BitTorrent and BitTorrent with ACCM

4.3 Algorithm overhead

In our prototype system, the access link congestion inferring algorithm will lead to some additional overhead which is the fundamental flaw in our system. This overhead directly proportional to our algorithm parameters *K* and *T*, and the two parameters will directly affect the accuracy of the algorithm. In our experiments, *K* is set to 10 and T is set to 1s, and the overhead is 0.8 KB/s. From the experiment results show at 4.2 and 4.3, we believe that the additional overhead caused by access link congestion inferring algorithm is in the acceptable range and will not degrade the performance of file downloading.

5. Conclusion and future work

In this paper we propose and design an aggregate congestion control mechanism called *ACCM* which enforces the friendliness of network upon all the connections belongs to P2P file sharing application instead of upon individual connection. *ACCM* adopts application-level measurement technology to infer network congestion situation, and on the basis uses AIMD algorithm to control parallel data transfer connections, achieving friendliness to the network and maximize utilization of network bandwidth. Furthermore, compared to other existing algorithms, *ACCM* needs neither the support of network nor the modification of protocol stack, so it can be easily implemented and deployed. The simulation experiments demonstrate that the certain fairness and congestion avoidance can be achieved in presence of congestion and the network bandwidth can be effectively utilized in absence of congestion with *ACCM*.

Our future work will focus on finding more accurate and cost less access link congestion inferring algorithms and evaluating the performance of other connection window control strategies such as MIAD, MIAD and MIMD.

6. References

[1] http://bitconjurer.org/BitTorrent/. December 2007

[2] http://www.kazaa.com/. December 2007

[3] CacheLogic Research. "The true pictures of P2P file sharing". 2004. http://cachelogic.com/research/slide1.php

[4] A. Banchs, "User Fair Queuing: Fair Allocation of Bandwidth for Users," in Proceedings of IEEE Infocom, 2002.

[5] H. Balakrishnan, H. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts," in Proceedings of ACM SIGCOMM, Cambridge, MA, September 1999.

[6] D. E. Ott, T. Sparks, and K. Mayer-Patel, "Aggregate Congestion Control for Distributed Multimedia Applications," in Proceedings of IEEE INFOCOM, Hong Kong, China, March 2004.

[7] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," in Proceedings ACM SIGCOMM, Stockholm, Sweden, August 2000.

[8] M. Singh, P. Pradhan, and P. Francis, "MPAT: Aggregate TCP Congestion Management as a Building Block for Internet QoS," in Proceedings of IEEE International Conference on Network Protocols (ICNP'04), Berlin, Germeny, October 2004.

[9] T. J. Hacker, B. D. Noble, and B. D. Athey, "Improving Throughput and Maintaining Fairness Using Parallel TCP," in Proceedings of IEEE INFOCOM, Hong Kong, China, March 2004.

[10] Shao Liu, Milan Vojnovi´c, Dinan Gunawardena. "4CP: Competitive and Considerate Congestion Control Protocol." In Proceedings of ACM SIGCOMM, 2006.

[11] T. Tsugawa, G. Hasegawa, and M. Murata, "Background TCP data transfer with inline network measurement," in Proceedings of APCC, Oct. 2005.

[12] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," in Proceedings of ACM SIGMETRICS, San Jose, CA, June 2000.

[13] D. Katabi, I. Bazzi, and X. Yang, "A passive approach for detecting shared bottlenecks," in Proceedings IEEE Conf. on Comp. Comm. and Networks, Arizona, Oct. 2001.

[14] W. Cui, S. Machiraju, R. Katz, and I. Stoica, "SCONE: A tool to estimate shared congestion among internet paths," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-04-1320, 2004.

[15] M. Kim, S. Lam, T. Kim, Y. Shin, and E. J. Powers, "A wavelet-based Approach to Detect Shared Congestion," in Proceedings of ACM SIGCOMM 2004, Portland, Oregon, August 2004.

[16] V. Jacobson and M. J. Karels, "Congestion avoidance and control," in Proceedings of ACM SIGCOMM, 1988.

[17] http://xbtt.sourceforge.net/