Friendly P2P: Application-level Congestion Control for Peer-to-Peer Applications

Yaning Liu, Hongbo Wang, Yu Lin, Shiduan Cheng, Gwendal Simon

Friendly P2P: Application-level Congestion Control for Peer-to-Peer Applications

Yaning Liu[1,2], Hongbo Wang[1], Yu Lin[1], Shiduan Cheng[1]
[1]State Key Lab. of Networking and Switching Tech.
Beijing Univ. of Posts and Telecommunications, China

Gwendal Simon[2]
[2]Computer Science Department
Institut TELECOM - TELECOM Bretagne, France

***Abstract*—Peer-to-Peer (P2P) file sharing applications use multiple TCP connections between peers to transfer data. The aggressiveness and robustness of P2P technology remarkably improve transfer efficiency and network bandwidth utilization. However, while the network bottleneck link is congested, P2P applications tend to unfairly steal bandwidth from other traditional Internet applications (Client/Server mode), which deteriorates the performance of traditional Internet applications. The paper proposes a *friendlyP2P* system with new application-level approaches for congestion detection and avoidance to keep fairness between P2P traffic and traditional Internet traffic. *friendlyP2P*, which is friendly to ISPs, namely to Internet networks and traditional Internet traffic, detects network congestion via throughput measurements and alleviates network congestion by optimization of the number of P2P connections from the viewpoint of P2P users. *friendlyP2P* system requires neither network node support nor TCP modification, which makes it easy to deploy. Simulation experiments demonstrate that fairness and congestion avoidance can be achieved in presence of congestion, and network bandwidth can be effectively utilized in absence of congestion with *friendlyP2P* technology.**

***Index Terms*—Peer-to-Peer, Friendly, Congestion Detection and Avoidance, Network Measurement.**

I. Introduction

Applications relying on peer-to-peer (P2P) architectures have become massively popular: file-sharing or phone system [1] are the most famous. This trend is scheduled to continue because numerous projects based on P2P data exchanges are currently under development, for example video streaming [2] or Distributed Hash Tables (DHT) [3]. However the growth of P2P traffic, which has already been noticed in the past [4], raises new issues for Internet Service Provider (ISP). Among them, the number of simultaneous TCP connections handled by P2P applications is threatening other traditional Internet applications. Indeed cooperation between peers is commonly implemented with multiple concurrent TCP connections in file-sharing applications as well as in most recent video streaming systems. As a consequence, the ratio of TCP connections for applications such as HTTP or VoIP over the total number of TCP connections is smaller, so the part of the bandwidth these latter applications could use becomes weaker. In other words, the fair bandwidth sharing mechanism of TCP fails in guarantying a fair sharing among applications.

This work is supported by China National Natural Science Foundation(90604019, 60502037); China National 863 Program(2006AA01Z235); China National Grand Fundamental Research 973 Program(2006CB701306); New Century Excellent Talents in University (NCET-07-0109).

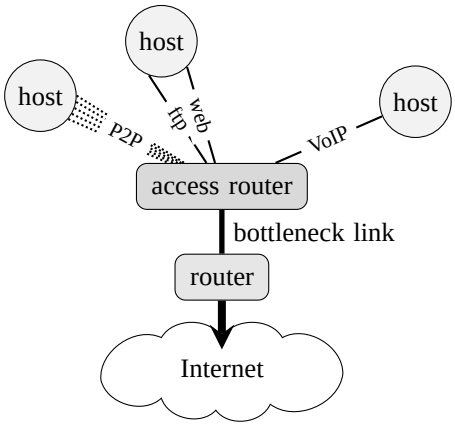

Fig. 1. Access point congestion: three hosts sharing an access point, dotted lines represent P2P connections and plain lines are for traditional applications

This observation could be neglected if the network bottleneck was far from the end users. Several studies have unfortunately shown that most network bottlenecks in the Internet are either in the access network or on the links between ISPs [5]. When congestion occurs on the shared access network which connects a xDSL router or a FTTH access router to the Internet, P2P applications with multiple TCP connections unfairly steal bandwidth from other traditional Internet applications. Moreover, the need for ubiquitous or pervasive Internet makes the number of devices connected to one access point, and consequently the number of applications served by this first router, increases dramatically. This concern advocates for an application-level fair sharing of bandwidth. This scenario is illustrated in Figure 1 where three hosts connected to Internet through an unique access point are running four applications using eight connections.

A. Limitations of Existing Techniques

In order to address the issue related with aggressive P2P applications, ISPs have defined a set of policies aiming at controlling P2P traffic within their backbone network. A first challenge consists in identifying this traffic [6], then to design a way to contravene it. Some technologies have been implemented [7], but they act on the backbone although the congestion occurs in the access network. Therefore they can probably not prevent congestion and, worst, they probably cut off some P2P connections for clients experiencing idle network

usage. It results that ISPs implementing such technologies may alleviate and even lose subscribers without actually improving the performance of their network.

Another approach requires the contribution of Internet routers. Integrated services (IntServ) based on RSVP protocol can be used to prioritize some flows, while differentiated services (DiffServ) defines a set of class of services which allows a traffic management based upon broad flow aggregates. Unfortunately these mechanisms require the cooperation of all routers. In a more recent work, flow-aware networking (FAN) [8] provides per-flow differentiation to active flows through implicit admission control and per-flow scheduling. FAN requires the association of the end user and its access router to solve the problem of the access network congestion, but the replacement and update of current access routers are costly. We would focus on mechanisms based on only end users without the support of any router.

In another approach motivated by distributed applications, "lower than best effort" mechanisms are especially useful for background transfer applications, so which may tolerate occasional throughput degradation, *e.g.* content prefetching and storage management in P2P systems. The idea is to infer network congestion in advance and back off earlier than loss-based TCP Reno. TCP Nice [9], TCP-LP [10] and 4CP [11] make the background transfer more sensitive to network congestion. Though they do not need any support from router, they require to modify the TCP protocol, which we can hardly assume. An application-level approach [12] has also been proposed to infer the available capacity and to adjust the sending rate of the background transfer by varying the receiver-advertised window size. This latter work is close to what we would like to do, but *friendlyP2P* realizes a real-time management of throughput which is actually far more variable than the usual one because the capacity of peers is very heterogeneous. Furthermore, P2P applications are designed so that a faulty connection is efficiently handled with self-stabilizing algorithms. On the contrary, experiencing a variable quality on a link can trouble most P2P algorithms where peers tend to aggregate based on their capacity.

B. Our contributions

We present in this paper *friendlyP2P*: an application-level congestion detection and avoidance which does not require neither router support, nor any TCP modifications. This system, intended to run on end users' devices, contains two components aiming at: (i) measuring the throughput of P2P flows to infer real-time status of the access network and (ii) relieving network congestion by adjusting the number of P2P connections accordingly. This second component may be implemented with regard to P2P applications, that is *friendlyP2P* could provide some network status and recommendations to other applications in order to let them adjust the number of connections by themselves. In this paper, we consider for simplicity that *friendlyP2P* can directly modify the number of flows associated with a P2P application.

The behavior of *friendlyP2P* is quite basic. As soon as the network is idle, *friendlyP2P* increases the number of P2P connections so that transfer efficiency and network utilization can be improved. Indeed, network bandwidth can be utilized at full steam in this case, so P2P applications can freely increase the number of connections. On the contrary, *friendlyP2P* changes the P2P traffic from aggressiveness to friendliness as soon as a network congestion is detected. The number of P2P flows is then reduced, so other applications may retrieve more bandwidth. This behavior is expected to alleviate network congestion, improve satisfaction of users and reduce the cost of network maintenance and capacity extension.

The paper gives a short description of this preliminary work. A simple model is presented in Section II. We then propose the *friendlyP2P* system in Section III. The focus of the description in this paper is restricted to fundamental ideas behind this system, especially algorithms for congestion control and congestion avoidance. Many more sophisticated approaches could be designed but we aim here to present basic but efficient algorithms because we emphasize the concept of *friendlyP2P* rather than its actual implementing details. Simulations demonstrate the validity of the *friendlyP2P* system in Section IV. Our goal is again to show that *friendlyP2P* can basically have a positive impact on access point network and to give an overview of the kind of results we may expect from preliminary works in this direction. Finally, future works and conclusive thoughts are given in Section V.

II. Model and Notations

The model described in the following concerns P2P applications using TCP protocol, *i.e.* P2P applications are assumed to be bulk transfer TCP flows. The packet loss rate p_i of one connection i is an indication of network congestion in TCP Reno and contributes also to the variability of TCP performance, especially the throughput noted $TH(p)$. The average Round Trip Time of the i^{th} TCP flow, denoted as RTT_i, equals to the sum of $T_{wait}(i)$ – the average waiting time in the queue of the bottleneck router – with $\tau(i)$ the propagation time determinated by the speed of light. The Maximum Size Segment MSS_i depends on the underlying network and operating system. We assume that MSS_i is identical and constant across all simultaneous TCP connections.

$$TH(p_i) = \frac{MSS}{RTT_i * f(p_i)} = \frac{MSS}{(\tau_i + T_{wait}(i)) \times f(p_i)}$$

The aggregated bandwidth of a P2P application using k simultaneous connections is BW. Following a well-known model for the steady state throughput of a bulk transfer TCP flow [13] and assuming that p_i and $T_{wait}(i)$ are equal for all P2P connections in the congested access network, we obtain:

$$BW \leq \frac{MSS}{f(p)} \left(\sum_{i=1}^{k} \frac{1}{\tau_i + T_{wait}} \right)$$

TCP congestion avoidance algorithm is an equilibrium process that attempts to balance all TCP flows to fairly share network bottleneck bandwidth. As the number of P2P flows

increases, the cross-traffic – web, VoIP and other Internet single-flow application – back off more to P2P traffic in the congested network bottleneck, which greatly improves the aggregated throughput of P2P traffic. A performance model of integrating P2P file sharing traffic and web traffic is proposed to quantify the impact of P2P traffic on web traffic in [14].

The *friendlyP2P* system takes place on a computer running simultaneously m P2P tasks generating n TCP connections. The number of connections managed by the i^{th} task is noted n_i which we consider as being variable between $MinNum$ and $MaxNum$, respectively the minimum and the maximum connection number of a P2P application. The throughput of the j^{th} connection of the i^{th} application is noted TH_{ij}. The network inferring mechanism is based on perpetual analysis of these connections. For simplicity, the notation we use for the last throughput measure and the congestion signal for this connection is noted $preTH_{ij}$ and $Cong_{ij}$ respectively. $CongFlag$ is the congestion signal as a whole sight depending upon all the P2P connections. Finally, γ and $\mu(\gamma \leq 1, \mu \leq 1)$ denote the additive increase and decrease factor respectively.

III. Friendly P2P system

With the *friendlyP2P* system, we seek to balance two conflicting goals: fairness and high network utilization. We assume that all applications are based upon the TCP Reno protocol which is the most popular TCP version presently. Congestion detection is based upon following ideas: in the absence of network congestion, TCP Reno increases its window by one, and then *friendlyP2P* will infer that many P2P flows are increasing their throughput and deem network idle. If TCP Reno detects network congestion, it halves its congestion window, so congested TCP flows halve their throughput. In this case, *friendlyP2P* will infer bottleneck router congested when many P2P flows halve their rates. We implement the congestion control algorithm of *friendlyP2P* on the download links where the impact of P2P traffic on the traditional Client/Server service is more severe. The study on upload link will be considered in the future work.

A. Congestion Detection Mechanism

The congestion detection mechanism has two components: throughput measurement and congestion detection. The throughput measurement from the viewpoint of the host is the precondition of the congestion detection algorithm. During data transferring, P2P applications commonly use *chunks* which are all the same size except for possibly the last one which may be truncated. We assume a P2P host keeps one or $more$ P2P connections with each peer to get all the chunks that the peer has. Based upon chunk transfer, we calculate the throughput of each P2P connection, TH_{ij} which is equal to the chunk size divided by a chunk transfer time. The algorithm for each P2P connection is stated in Algorithm 1. Algorithm sensitiveness may be adjusted through two control parameters α and β ($\alpha \geq 1, 0 \leq \beta \leq 1$). The more α is decreasing or β is increasing, the more sensitive is the algorithm to infer change of network status. Especially as $\alpha = 1$ or $\beta = 1$, *friendlyP2P* infers the network idle or congested as soon as current throughput measured is larger or less than previous one respectively.

Algorithm 1: Congestion detection for one P2P connection

```
if (TH_ij > α × preTH_ij) then
    preTH_ij = TH_ij;
    Cong_ij = 0;
else if (TH_ij < β × preTH_ij) then
    preTH_ij = TH_ij;
    Cong_ij = 1;
else
    Cong_ij = −1;
end
```

Algorithm 2 shows the algorithm for all P2P connections. If more than half P2P connections are increasing their throughput, which means an idle network, *friendlyP2P* sets $congFlag = 0$. If more than half P2P connections are decreasing their throughput, which means a congested network, *friendlyP2P* sets $congFlag = 1$. In other scenarios *friendlyP2P* infers the network steady and sets $congFlag = -1$.

B. Congestion Avoidance Mechanism

If $congFlag$ is null, *friendlyP2P* supposes the network is idle and increases the number of P2P connections. Rather, a $congFlag$ set to 1 means that access network should be congested, so *friendlyP2P* triggers the congestion avoidance algorithm. In other scenarios, the concurrent number of P2P connections is kept unchanged.

The initial number of P2P connections is noted n_0. In case of idle network, *friendlyP2P* increases the number of connections by the factor γ for each task until inferring a stable (or congested) network or reaching $MaxNum$. When network is congested, *friendlyP2P* decreases the number of connections by the factor μ for each task until inferring a stable (or idle) network or reaching $MinNum$. As γ and μ increasing, the network status will change acutely. We set the default value of $MinNum$ to 1 to guarantee at least one TCP stream's fair portion of network bandwidth. The congestion avoidance algorithm is described in Algorithm 3.

IV. Performance Evaluation

In this section, we evaluate the performance of *friendlyP2P* system in several scenarios. Our objective is to explore the behavior of download links in the bottleneck link - the access network. We define *generalP2P* as a basic P2P file sharing application, where a peer gets from many peers some chunks. On the contrary, *friendlyP2P* is the same application augmented with the application-level congestion control. We compared the impact of *friendlyP2P* traffic with *generalP2P* traffic on the Internet traditional traffic, including FTP, UDP and HTTP cross-traffic.

We use NS-2.24 and the topology in Figure 2. GnutellaSim [15] is a scalable packet-level Gnutella simulator. By modifying the application layer and the protocol layer, we

Algorithm 2: Congestion detection for all P2P connections

if ($\frac{1}{2}\sum_{i=1}^{m} n_i >$ (*the number where* $Cong_{ij} = 1$)) **then**
  $congFlag = 0$;
else if ($\frac{1}{2}\sum_{i=1}^{m} n_i >$ (*the number where* $Cong_{ij} = 0$)) **then**
  $congFlag = 1$;
else
  $congFlag = -1$;
end

Algorithm 3: Congestion avoidance algorithm

for *i=1* **to** *M-1* **do** $n_i = n_0$;
for *i=1* **to** *M-1* **do**
  if ($congFlag = 0$ && $n_i < MaxNum$) **then**
    $n_i = n_i + \gamma$;
  else if ($congFlag = 1$ && $n_i > MinNum$) **then**
    $n_i = n_i - \mu$;
  else
    n_i unchanged;
  end
end

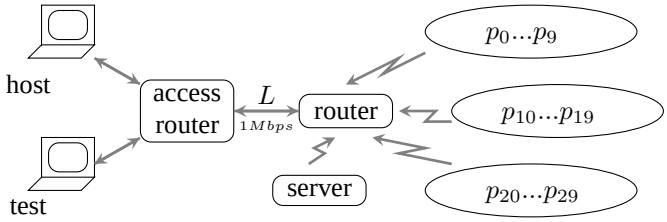

Fig. 2. The simulation topology

extend GnutellaSim to support congestion control of *friendlyP2P*. We run *friendlyP2P* on peers and the P2P end user $host$ that is the measurement point in the access network. Cross traffic are generated from peers to $test$. The bottleneck link L is the export of the last-mile access network with drop-tail FIFO queuing and is set to $1Mbps$. Other links are set to $100Mbps$ to make the link L be the only bottleneck. The buffer size is set to the bandwidth delay product. Packets are set to 512 bytes in size and the propagation delays are set to a random value in interval $(10ms, 100ms)$.

We focus on the performance of one end user, thereafter called $host$, in a congested access network. The number of simulated peers does not require to be high, because our concern is to build as many P2P connections as possible to congest the bottleneck. The peers are denoted $P_0, P_1, ..., P_{29}$. After running the experiment several times and comparing the experiment result, we basically set $\alpha = 2, \beta = 1/2, \gamma = 1, \mu = 1$ in Alg. 1 and Alg. 3. We will evaluate the exact tuning of these parameters in future works.

A. Competing with FTP traffic

We first consider that P2P traffic coexists with FTP traffic. Five P2P tasks are run between $host$ and its peers and five FTP tasks between $test$ and $server$ respectively. Each P2P task has four TCP connections and FTP task has one TCP connection. So 20 P2P flows are run between $host$ and peers and 5 FTP flows are run between $test$ and $server$.

Figure 3 and Figure 4 compare temporal dynamics of the aggregated throughput of FTP traffic with *generalP2P* traffic and *friendlyP2P* traffic respectively. Figure 3 shows that *generalP2P* traffic is aggressive in taking most of the bottleneck bandwidth. Through dynamically inferring network status, *friendlyP2P* can be modest and keep the fairness between P2P traffic and FTP traffic in Figure 4.

B. Competing with UDP traffic

The interaction of *friendlyP2P* traffic with an UDP flow is investigated in this section. The rate of the UDP flow increases from 200Kbps to 800Kbps when the time is 3000 second. Figure 5 and Figure 6 show the throughput of *generalP2P* traffic and *friendlyP2P* traffic respectively.

Aggregated throughput of *friendlyP2P* in Figure 6 fluctuates more drastically than *generalP2P* in Figure 5, because congestion control algorithms make throughput of *friendlyP2P* traffic fluctuate according to current network status. When bottleneck network is congested, *friendlyP2P* back off, which makes UDP traffic change more smoothly in Figure 6. Especially when UDP traffic violently increases from 200Kbps to 800Kbps, UDP traffic changes more drastically in Figure 5. Experiment results show that *generalP2P* is more aggressive than *friendlyP2P* while coexisting with UDP traffic.

C. Competing with WEB traffic

We also explore the impact of the number of P2P connections on web latency which means duration of getting a Web page. For performance idealization and simplicity, web latency is investigated in a scenario where only one web-traffic flow exists. Web traffic is run from $server$ to $test$. With the same direction as web traffic, P2P traffic is from $(p_0, ..., p_{29})$ to $host$. $MaxNum$ of *friendlyP2P* equals the number of *generalP2P* in Figure 7.

When the number of P2P connections are not big enough to make the network congested, web latency in *friendlyP2P* scenarios almost equals that of *generalP2P* scenarios. As the number of P2P connections increases, network congestion occurs. *friendlyP2P* adjusts the number of P2P connections to alleviate network congestion. With congestion control of *friendlyP2P*, a significant improvement in the performance of the web traffic is obtained in presence of network congestion.

D. Network utilization

We compare network utilization between *generalP2P* and *friendlyP2P* in Figure 8. Before the time of 170s, there is only P2P traffic in the network. Then we inject Web and FTP traffic into the network. At the beginning of the experiment, the aggregated throughput of *generalP2P* increases faster

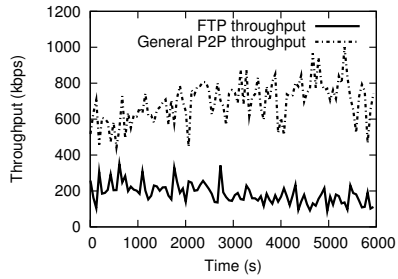

Fig. 3. Comparing the aggregated throughput between *generalP2P* traffic and FTP traffic

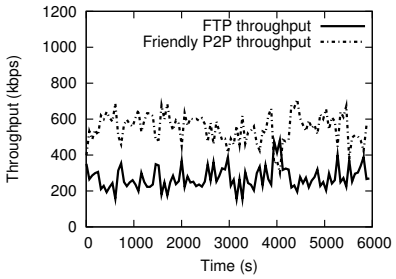

Fig. 4. Comparing the aggregated throughput between *friendlyP2P* traffic and FTP traffic

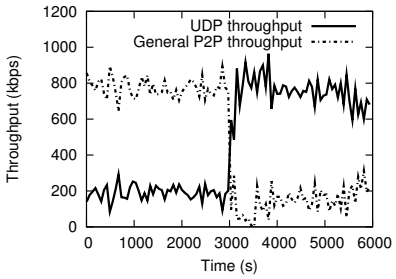

Fig. 5. Comparing the aggregated throughput between *generalP2P* traffic and UDP traffic

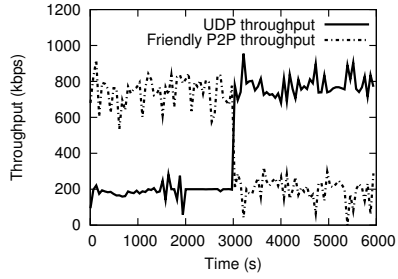

Fig. 6. Comparing the aggregated throughput between *friendlyP2P* traffic and UDP traffic

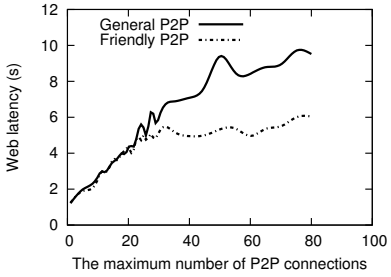

Fig. 7. Web latency versus the number of P2P connections

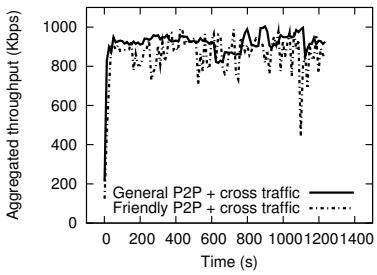

Fig. 8. Comparing the throughput for two kinds of P2P traffic

than *friendlyP2P*, because *friendlyP2P* additively increases the number of connections to alleviate the impact on the Internet network and traditional Internet traffic. The dotted curve after 50s shows *friendlyP2P* can keep the same high network utilization as *generalP2P*. After 170s, the aggregated throughput changes more drastically in the *friendlyP2P* scenarios, which shows that congestion control algorithms of *friendlyP2P* can work well as the access network is congested and can also keep high average network utilization.

V. Conclusion

This paper presents *friendlyP2P*: a congestion control algorithm designed to alleviate the impact of P2P traffic on traditional Internet traffic. This proposal aims to allow P2P applications to benefit from an intensive network utilization in the absence of network congestion, but to also switch to a more friendly mode as soon as a network congestion occurs. This preliminary proposal is appropriate for any P2P system where each peer gets some fixed-size chunks from many peers through many TCP connections. One strong point behind *friendlyP2P* is that it requires neither support of routers nor any TCP modification, so it can be easily deployed in the Internet. The NS2 simulations validate the effectiveness of *friendlyP2P*. Compared to a basic P2P application, *friendlyP2P* can improve the performance of traditional Internet traffic when network congestion occurs. We believe mutual benefit among ISPs, P2P users and non-P2P users can be achieved through the *friendlyP2P* solution. Future works will extend this paper, especially more complex congestion detection mechanisms will be defined and all parameters which may impact the system behavior will be precisely evaluated.

References

[1] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *IEEE Infocom*, 2006.

[2] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale p2p iptv system," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[3] S. Rhea, B. Godfrey, B. Karp, and J. Kubiatowicz, "Opendht: a public dht service and its uses," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 73–84, 2005.

[4] C. Research, "The true pictures of p2p file sharing," 2004. [Online]. Available: http://cachelogic.com/research/slide1.php

[5] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *ACM Conf. on Internet Measurement*, 2003.

[6] F. Constantinou and P. Mavrommatis, "Identifying known and unknown peer-to-peer traffic," in *IEEE Int. Symposium on Network Computing and Applications*, 2006, p. 93–102.

[7] [Online]. Available: http://www.p-cube.com/index.shtml

[8] S. Oueslati and J. Roberts, "A new direction for quality of service: flow-aware networking," *Next Generation Internet Networks*, April 2005.

[9] A. Venkataramani, R. Kokku, and M. Dahlin, "Tcp nice: A mechanism for background transfers," in *Operating Systems Design and Implementation (OSDI'02)*, 2002.

[10] A. Kuzmanovic and E. W. Knightly, "Tcp-lp: low-priority service via end-point congestion control," *IEEE/ACM Trans. Netw.*, 2006.

[11] S. Liu, M. Vojnovic, and D. Gunawardena, "4cp: Competitive and considerate congestion control protocol," in *ACM SIGCOMM*, 2006.

[12] P. Key, L. Massoulie, and B. Wang, "Emulating low-priority transport at the application layer: A background transfer service," in *ACM SIGMETRICS/Performance'04*, June 2004.

[13] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, 2000.

[14] Y. Liu, H. Wang, Y. Lin, and S. Cheng, "Modeling and quantifying the impact of p2p traffic on traditional internet traffic," in *22nd Int. Conf. on Advanced Information Networking and Applications Workshop (AINAW'08)*, 2008.

[15] He, Q. Ammar, M. R. G. Raj, and H. Fujimoto, "Mapping peer behavior to packet-level details: a framework for packet-level simulation of peer-to-peer systems," in *11th IEEE/ACM International Symposium on MASCOTS 2003*, 2003.