TOKEN-BASED CONGESTION CONTROL: ACHIEVING FAIR RESOURCE ALLOCATIONS IN P2P NETWORKS

Zhiqiang Shi

Institute of Software, Chinese Academy of Sciences
zhiqiang@ios.cn

ABSTRACT

Fair Queueing designed to achieve fair bandwidth allocations like CSFQ and Stochastic Fair BLUE, have many desirable properties for congestion control in Internet. However, such mechanisms usually supervise the bandwidth consumption of per-flow, and are good for nothing while P2P flows dominate the traffic of Internet. In this paper, we propose a Token-Based Congestion Control, which limits the access token resource consumed by every subscriber, and provides substantial fairness in P2P networks. In this congestion control system, there are three important devices: core routers, edge routers, terminals. Core routers measure congestion level, and convey it to terminals along with data flows. Terminals label the Token-Level on sent packets according to the Congestion-Index in the transport path, and regulate the average speed of output tokens, which are the multiplication of the packet size and the Token-Level. Edge routers police the input token rate of every terminal. We present simulations and analyses on the performance of this approach at last.

***Keywords*—**Congestion Control, P2P, CSFQ, Token-Level, congestion level, congestion index.

1. INTRODUCTION

The Peer-to-Peer (P2P) Computing allows users to make use of the collective power in the network, overcomes the limitation of the centralized or hierarchical service mode, which have been applied to File Exchange, Cooperative Work, Search Engine, and Instant Communication. According to the report of CacheLogic, P2P traffic exceeding Web traffic, was 60% (and rising) of Internet traffic in 2004, with Bit-Torrent [1] accounting for 30% of traffic. While Bit-Torrent speedups the construction of broadband networks, it also absorbs magnitude bandwidths by immense connections, which results in low throughput of each connection, and put impetuous impact on traditional network applications, and bring the huge challenge to TCP congestion control and traditional fair queueings.

This work is supported by Chinese Next Generation Internet under Grant No.2006-2-5, and by Chinese National 973 Fundamental Research Program under Grants No.2007CB307100 and No.2007CB307106.

Fairness among Flows

To share network resources among competitive flows, we have so far mostly relied on pure end-to-end mechanisms, in particular TCP Congestion Control. After Random Early Discard (RED) [2] has been introduced, router begins to regulate the behaving of total traffic to improve performance, such as more throughputs and less queueing delays. Until Core-Stateless Fair Queueing (CSFQ) [3] is presented, fair queueings were typically achieved by using per-flow queueing or per-flow dropping such as Weighted Fair Queuing (WFQ) [4] and Flow Random Early Drop (FRED) [5], which are too complex to implement for the best-effort traffic.

In order to approximate fair queueing while limiting flow state to the edge of network and removing it from core routers, CSFQ inserts state information to the packet header at edge router. The *flow* defined in CSFQ is the integration of all the packets which follow the same path within the core. The flow can be defined as a source-destination pair or source-destination-port pair. According to the routing scheme of OSPF and RIP, the router can only retrieves the next hop address, so the granularity of flow should be no more than source-destination pair. If the flow is defined as source-destination-port pair, the multi-threading download can steal more bandwidth than the single-threading application. If the flow is defined as source-destination pair, Bit-Torrent can grab more bandwidth in the networks deployed with the CSFQ. There are many improvement algorithm of CSFQ [6][7], but all of them fail to recognize the revolution made by the P2P Computing.

Internet with P2P Computing

Bit-Torrent is a new file exchange method, which is different from the traditional Client/Server mode. In Bit-Torrent, the file downloading is transferred directly from other clients, the idea behind it is all for one and one for all. The fact behind the technique is that a task of Bit-Torrent is usually serviced by tens of such flows, while the traditional application is serviced by only one or two flows. The largest unfair is that CSFQ only provides fair to each flow, but not to individual application or subscriber.

The flow fairness inspirits users and software providers to setup more connections for grabbing the extra throughput, and then more flows travel through congestion links. According to TCP friendly Equation (1) [8][9]

$$Bandwidth = 1.3 \times MTU / (RTT \times \mathbf{sqrt}(Loss)) \quad (1)$$

Where the *Bandwidth* is the average throughput of the TCP flow, the *MTU* is the average packet size, the *RTT* is the average round trip delay and the *Loss* is the loss rate being experienced by the TCP flow. It is easy to transfer the equation (1) into equation (2)

$$Loss = \mathbf{square}(1.3 \times MTU / (Bandwidth \times RTT)) \quad (2)$$

As the total bandwidth of the congestion link is constant, according to the equation (2), the more flows transport the congestion link, the less bandwidth each flow can occupy, and the higher the loss rate is. CSFQ is not only unfair among user and application, but also brings poor performance into networks dominated by Bit-Torrent traffic.

Fairness with P2P Computing

In this paper, we first grade each packet with a Token-Level label at the sender according to the Congestion-Index. The higher the congestion level of a link is, the more congestion the link is; the Congestion-Index is the maximum of the congestion level of the links in the transport path. We define a new type of resource called *token*, the amount of tokens consumed by a packet are the multiplication of the packet size and the packet's Token-Level. Next, we limit all the amount of tokens consumed by each subscriber. When networks are in congestion, the Congestion-Index increases, and the Token-Level of incoming packets increases too. Considering the access token resource of a subscriber is constant, the traffic will slow down, which is our congestion avoidance algorithm.

This paper is organized as follows. In section 2, we propose our Token-Based Congestion Control (TBCC) scheme in detail. Section 3 gives the experimental results. In last section, we summarize this paper. All the experiments in this paper are performed on network simulator ns2 [10].

2. THE ARCHITECTURE OF TBCC

The same as CSFQ, there are three important devices core routers, edge routers and intelligent terminals in the TBCC, connected up as Figure 1. Different from CSFQ, the congestion information of TBCC is conveyed from the network to the user in the packet extend header *tkheader*, which include the labels of *tklevel*, *tkpath* and *tkback*. The *tklevel* represents the Token-Level of the packet, the *tkpath* used to collect the highest congestion level in the transport path of the packet, the *tkback* used to return the Congestion-Index of the forward path to the peer node. In the paper, the *tklevel*, *tkpath*, and *tkback* range as an integer from 10 to 100.

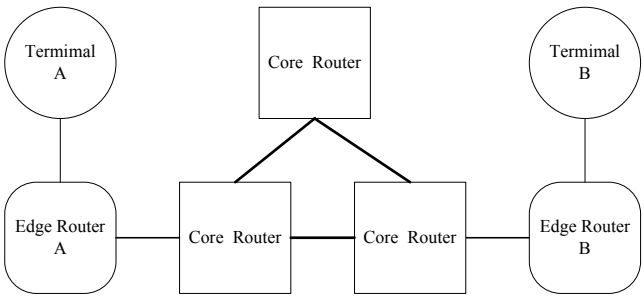


Figure 1: The architecture of TBCC

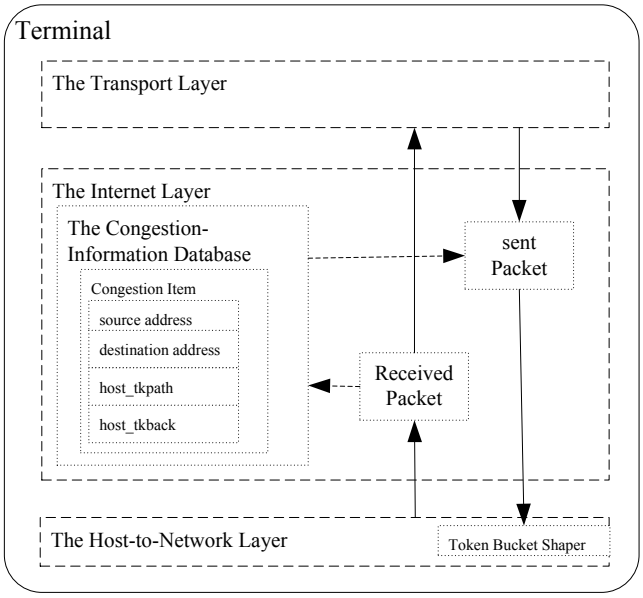


Figure 2: The intelligent terminal

Intelligent Terminals

The detail of the intelligent terminal is described in Figure 2. There is a congestion information database (*CID*) in the Internet Layer of every terminal, and each congestion item is composed of a source address, destination address, *host_tkpath* and *host_tkback*, which is indexed by the two elements of the source address and destination address. Congestion information is updated while receiving packet and is labeled onto sent packets, the detail process is as follow. When a packet is received, *CID* picks up the source and destination address from the packet, locates the corresponding congestion item, and replaces *host_tkpath* and *host_tkback* with *tkpath* and *tkback*. When a packet is sent, *CID* picks up the source and destination address, locates the corresponding congestion item, and labels the *tklevel*, *tkback* and *tkpath* with *host_tkback*, *host_tkpath* and 10.

There is also a token bucket shaper in the output interface of the terminal. We set the unit of Congestion-Index as *ci*, the unit of token as *tk*, which is the multiplication of *bit* and *ci*. The carrier and subscriber make an access agreement on the token bucket algorithm with the token capacity as C *tk* and the token arrival rates as ρ *tk/sec*. The counter of the token in the token bucket shaper is ξ (*tk*). The packet can be sent to the edge router, only if the counter ξ is great than zero. After the sent of a packet with the total length L *bit* and the Token-Level γ *ci*, the counter ξ should decrease $L * \gamma$.

Edge Routers

There is also a token bucket controller at the input port of the edge router as Figure 3. The token bucket controller is configured with the token capacity and the token arrival rates the same as the token bucket shaper. When the edge router receives a packet and the counter of the token is negative, the packet will be dropped. On the contrary, the packet will be transferred to networks, and the counter ξ should decrease $L * \gamma$.

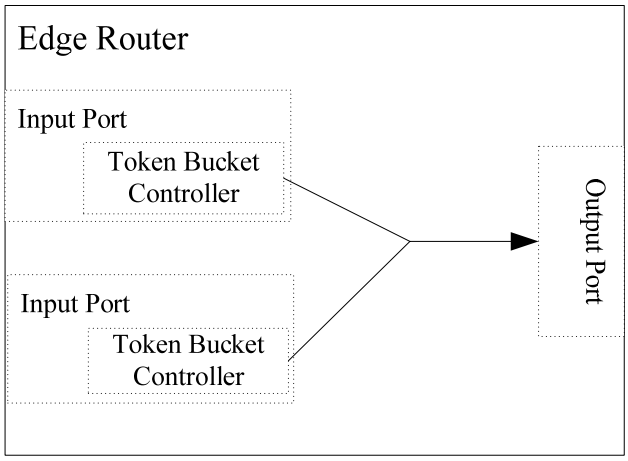


Figure 3: The edge router

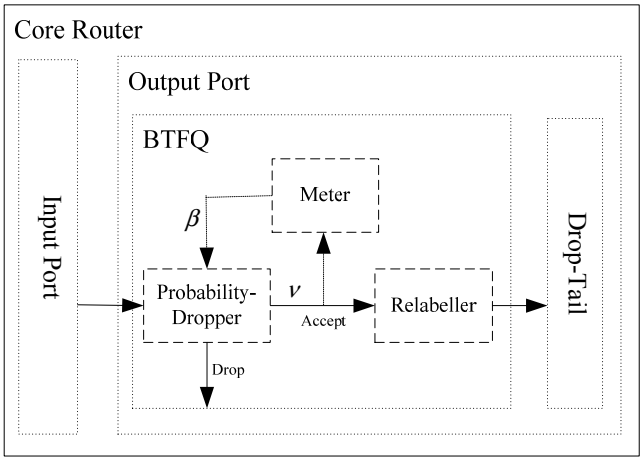


Figure 4: The core router

Core Routers

There are Bit-Torrent Fair Queueing (*BTFQ*) and Drop-Tail at the output port of the core router as Figure 4. *BTFQ* consists of a Probability-Dropper, Meter and Relabeller. The congestion level β is an important parameter of *BTFQ*, which ranges as an integer from 10 to 100 in this paper. If β is 10, we say the link is idle. The higher the congestion level is, the more congestion the link is.

The congestion level β is used in the Probability-Dropper, and adjusted by the Meter. When the Probability-Dropper receives a packet with Token-Level γ, it compares the congestion level and the Token-Level. If the Token-Level is larger than or equal to the congestion level, it receives the packet. On the contrary, it calculates the fair drop probability *prob* of the packet according to equation (3)

$$prob = 1 - \omega \times \gamma / \beta \tag{3}$$

and drop the packet with the probability *prob*, where the ω is a punish parameter less than 1. When the Relabeller receives the output packet of the Probability-Dropper, it relabels the *tklevel* and *tkpath* of the packet according to equation (4)

$$\begin{cases} \textbf{if } tklevel < \beta, tklevel = \beta; \\ \textbf{if } tkpath < \beta, \ tkpath = \beta; \end{cases} \tag{4}$$

As the output link speed S of the output port is constant, the Meter can adjust the congestion level to driver the output packet rate ν approaching to the link speed S. If the output rate ν is persistently larger than the link speed, the congestion level should increase, on the contrary, the congestion level should decrease. The updating algorithm is described by equation (5), where the β is the current congestion level of BTFQ, and the β' is the updated one.

$$\beta' = \beta * \nu / S \tag{5}$$

The Control System of TBCC

We summarize the difference of CSFQ and TBCC, and list it in Table 1. In CSFQ, intelligent terminals do not have any additional function. Edge routers should classify the input traffic into individual flows, measure the input speed of each flow, and label it to the packet. Core routers measure the congestion of output link, estimate the fair bandwidth, drop the extra packets in flow, and relabel the speed of output packets.

Table 1: The comparison of CSFQ and TBCC.

Device	CSFQ	TBCC
Intelligent Terminal	—	Label Shape
Edge Router	Classify Measure Label	Police
Core Router	Measure Drop Relabel	Measure Drop Relabel

In TBCC, intelligent terminals label the Token-Level on sent packets according to the Congestion-Index of the transport path, and shape the traffic to be consistent with the contract. Edge routers maintain the state of per subscriber, and police the consumed tokens of every subscriber, which is less complexity than CSFQ. Core routers use FIFO packet scheduling to measure the congestion of the output link, estimate the congestion level of their output links, drop the extra packet of the subscriber, and relabel the *tklevel* and *tkpath* of the output packet, which is the same complexity as CSFQ.

The control system of TBCC is distinct from CSFQ. In CSFQ, the fair bandwidth is not conveyed to the intelligent terminal directly, the TCP flow adjusts it's sent speed according to the loss rate, the non-responsive flow ,such as some UDP flows, sends traffic at constant rate to networks. As the edge router in CSFQ accepts all the input packets, the core router have to drop a large amount of extra packets. In TBCC, the Congestion-Index is conveyed to the intelligent terminal in the back-channel, all the flows label packets with corresponsive Token-Level, the edge router limits the input traffic according to the congestion level of networks, and there are only few extra packets dropped at core routers. The edge router could further cache the Congestion-Index in the back-channel, and force the Token-Level of the input packets to be consistent with the

cached information. It is apparent that TBCC overwhelms CSFQ in the performance.

Fairness of TBCC

TBCC is fairness to every subscriber, the more access token resource the flow has, the more throughputs the flow can achieve.

Theorem 1: For a flow with limited access token resource, there is an unique optimal solution to label the Token-Level of sent packet for achieving best throughput, which is equal to the *tkback* in the back-channel.

Proof: We assume the allowed input token rate of the flow f is ρ, the Congestion-Index of the flow f is γ. If we label the Token-Level of the sent packet with γ, the maximum input rate of the flow is ρ/γ at edge router. Thinking about the Token-Level γ is larger than or equation to the congestion level of all the core routers in the transport path, there should be no packet dropping at all the BTFQ in the path, so the throughput of the flow is about ρ/γ. If we send packets with Token-Level γ' larger than γ, the throughput of the flow is about ρ/γ', which is less than ρ/γ.

If we send packets with Token-Level γ'' less than γ, the maximum input rate of the flow is ρ/γ'' at edge router. Assumption that the maximum congestion level is at router $n(k)$, and there are k core routers, the sequence number of which are $n(1)$, $n(2)$, ⋯, $n(k)$, the congestion levels of which are maximum from the source to itself. Then, the following equation exist:

$$n(1) < n(2) < ... < n(k-1) < n(k)$$

$$\beta[n(1)] < \beta[n(2)] < ... < \beta[n(k-1)] < \beta[n(k)]$$

According to the equation (3), the throughput of the flow is $\omega^k \times \rho / \beta[n(k)]$, it is else equal to $\omega^k \times \rho/\gamma$, which is less than ρ/γ, because ω is the punish parameter , and less than 1.

So the unique optimal solution is equal to the Congestion-Index of the path, which is also equal to the *tkback* in the back-channel.

Deduction 1: For a Bit-Torrent application with limited access resource, it can achieve better throughput, but do not hurt the performance of networks.

Proof: We assume the allowed input token rate of a Bit-Torrent application APP in a subscriber is ρ, the Bit-Torrent APP have two distinct flows $f1$ and $f2$, and the Congestion-Index of flows $f1$ and $f2$ are $\gamma 1$ and $\gamma 2$, where the $\gamma 1$ is larger than the $\gamma 2$.

We discuss three schemes allocating the token resource of the Bit-Torrent. The first allocates the token resource equally between the two flows, the token resources of the flows $f1$ and $f2$ are both equal to $\rho/2$. The second allocates the token resource inverse proportion to the Congestion-Index of the flow, the token resources of the flows $f1$ and $f2$ are $\rho \times \gamma 2/(\gamma 1+\gamma 2)$ and $\rho \times \gamma 1/(\gamma 1+\gamma 2)$. The last is allocating all the token resource to the less congestion flow, the token resources of the flows $f1$ and $f2$ are 0 and ρ.

According to Theorem 1, the throughput of the application APP is $\rho \times (\gamma 1+\gamma 2)/(2 \times \gamma 1 \times \gamma 2)$ in the first allocation scheme, $\rho \times (\gamma 1/\gamma 2 + \gamma 2/\gamma 1)/(\gamma 1+\gamma 2)$ in the second, and $\rho/\gamma 2$ in the last. The relationship is as following.

$$\rho/\gamma 2 > \rho \times (\gamma 1/\gamma 2 + \gamma 2/\gamma 1)/(\gamma 1+\gamma 2)$$

$$\rho \times (\gamma 1/\gamma 2 + \gamma 2/\gamma 1)/(\gamma 1+\gamma 2) > \rho \times (\gamma 1+\gamma 2)/(2 \times \gamma 1 \times \gamma 2)$$

Based on the above analysis, the more token resources are allocating to low Congestion-Index flow, the more throughput the Bit-Torrent can achieve. It will also decrease the congestion level at congestion link, increase the congestion level at the idle link and at last make the traffic of networks more even. As to the Bit-Torrent Application with a lot of flows, apparently the conclusion is right too.

3. THE EXPERIMENTAL RESULT

In the section, we evaluate and demonstrate the performance and fairness of TBCC by simulations.

Amendment to TBCC

In TBCC, the packet loss at edge router is totally different to the packet loss at core router, the former is the congestion signal for sources to decrease the output rate, and the latter is an adjustment signal of the core router, which does not demand the source shrinks. As TCP flows are acute to the packet loss and can not distinguish between the two type of packet loss, the Token-Level of sent packet had better be a little higher than the Congestion-Index to avoid the packet drop at core routers for the vibration of the congestion level. In the paper, we set the Incremental to the Congestion-Index as 2.

Many subscribers are business users, and have a lot of intelligent terminals to share the same output link. The token resources should be allocated among terminals by subscriber gateways according to the service rule. As burst is inbeing of TCP flows, the shaper function in terminal should be shifted into gateways. In the paper, we adopt Start Potential-based Jitter Bounded Queueing (SPJBQ) [12] to implement the shaper at the terminal and gateway, which is improved to schedule the queues by token resources. A additional parameter η called *enlarger* is enhanced to SPJBQ. The allowed input token rate of a subscriber is equal to the multiple of the *enlarger* and the access bandwidth.

At core routers, the Drop-Tail may drop packets for overflow, which is also a signal of temporary congestion. The temporary congestion maybe caused by the burst of TCP flows, or maybe caused by temporary load augment. For quick adapting to load modification and the burst of TCP flows, we increase the congestion level of TBFQ a little step, when there is a packet drop at the Drop-Tail.

Thinking over the oscillation of the TCP flow, the bandwidth in core routers should divide into two parts to decrease the packet drop of TCP flow at Drop-Tail. The

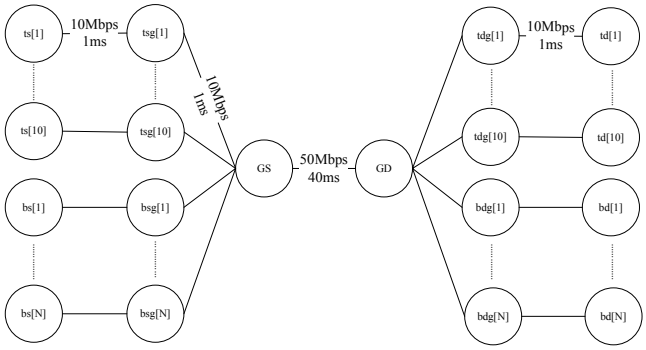


Figure 5: The topology for single congestion link scheduled by FIFO, RED, FRED, CSFQ, and TBCC

first part scheduled with high priority is used to the packets with token label, the other is used to the packet without token label, and the ratio of high priority bandwidth in the total bandwidth is τ.

Because there is the delay between the increasing of the congestion level in core routers and the response at terminals, the increment of the congestion degree should be smooth. So, we adjust only half step of the equation (5), which is just as equation (6). It is asymptotic to the theoretical value of the congestion level.

$$\beta' = \beta + (\beta * \nu / (S \times \tau) - \beta) / 2 \tag{6}$$

Single Congestion Link

To provide some context, we compare TBCC's performance of average throughput to four additional queueings in a single congestion link. Two of them are FIFO and RED, represent baseline cases where routers do not attempt to achieve fair bandwidth allocations. The other two queueings, FRED and CSFQ, represent different approaches to achieving fairness on each flow.

These four algorithms represent four different levels of complexity. FIFO is the least cost, the next is RED. FRED have to classify incoming flows, whereas FIFO and RED do not. CSFQ edge routers have complexity comparable to FRED, and CSFQ core routers and have complexity comparable to RED. TBCC core routers have the same complexity as CSFQ core routers, TBCC edge routers have the same complexity as RED. Although the TBCC in this paper need the collaboration of terminals, all the functions in terminals and gateways can shift into edge routers.

All simulations were performed in ns-2, which provide accurate packet-level implementation for various network protocols, and various buffer management and scheduling algorithms. All algorithms used in the simulations except TBCC, were part of the standard ns-2 distribution and [3].

We use the topology as Figure 5 to simulate the single congestion link. the bandwidth of access link is 10Mbps, and the link delay is 1ms. There are edge routers of tsg[k], tdg[k], bsg[i] and bdg[i] between intelligent terminals and core routers, where the k ranges from 1 to 10, the i ranges from 1 to N. The links between the intelligent terminals and the edge routers are scheduled by SPJBQ, which play as a token bucket shaper to avoiding the packet drop at the edge routers, the bandwidth is 10Mbps, the link delay is 1ms, and the enlarger is 10. The ratio of high priority bandwidth in the link from GS to GD is 90 %.

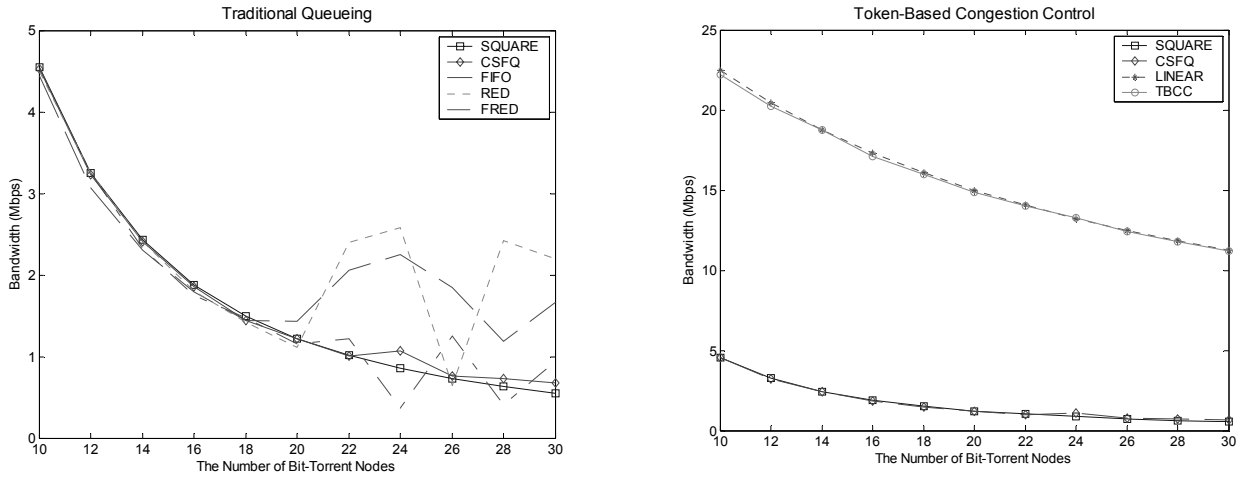
In Figure 5, there are ten traditional network applications competing with a P2P application of 2N nodes, where N ranges from 10 to 30. The traditional network applications send packets from the node ts[k] to td[k], the k ranges from 1 to 10. The P2P application sends packets from source bs[i] to destination bd[j], the i and j range from 1 to N, the number of total flows in the P2P application are N^2. The single congestion link is from node GS to GD, the bandwidth is 50Mbps, the link delay is 40ms.

As to the FIFO, RED, FRED and CSFQ, they all try to allocate the same bandwidth to each flow, in which the average throughput of the ten traditional network applications should decline squarely along with the increase of the P2P nodes N. As to TBCC, it try to allocate the bandwidth corresponding to the access token resource, the average throughput of the ten traditional network applications should decline linearly along with the increase the P2P nodes N.

The first simulator of single congestion link is that the ten traditional unresponsive UDP applications compete with a P2P application of 2N nodes based on unresponsive UDP, where all the UDP flows sent packets in CBR mode at the speed 500Kbps with random parameter true. Figure 6 shows the average throughput of the ten UDP applications by five different queueings. In Figure 6(a), there are five curves, the SQUARE represents the theory throughput of the ten traditional UDP application, the FIFO, RED, FRED and CSFQ represent the throughputs simulated by the corresponding queueings. According to Figure 6(a), the throughput variations of the four simulator results are consistent with the SQUARE while the number of P2P nodes is less than 20. When the number of P2P nodes N is large than 20, the output links from intelligent terminal to edge router are in full speed, which impels the simulator results of FIFO, RED and FRED departing from the SQUARE for the global synchronization. At the same time, the curve of CSFQ still adheres to the SQUARE, which demonstrates that the performance of CSFQ is better than the former three queueings.

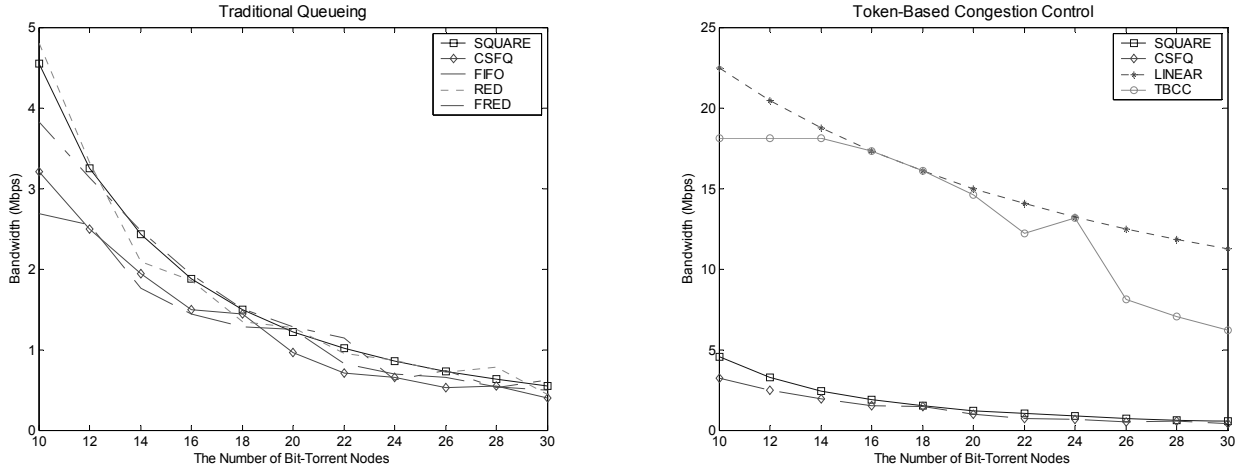
In Figure 6(b), the curve of TBCC represents the average throughput of the ten UDP applications, which is adhering to the LINEAR curve, while the curve of CSFQ is adhering to the SQUARE curve. It demonstrates that the TBCC gives better fairness to the traditional UDP applications than the CSFQ.

The second simulator of single congestion link is that the ten traditional TCP applications just like FTP, compete with a P2P application of 2N nodes based on TCP just link Bit-Torrent. In Figure 7(a), the four curves of FIFO, RED, FRED and CSFQ are all adhere to the curve of SQUARE, which shows the four queueings can provide good fairness among TCP flows. In Figure 7(b), the average throughput is bias from the curve of LINEAR in two cases. The first is the number small than 14. As there is small number of unavoidable packet drop at the Drop-Tail of core router, according to the equation (1), the access token resource can



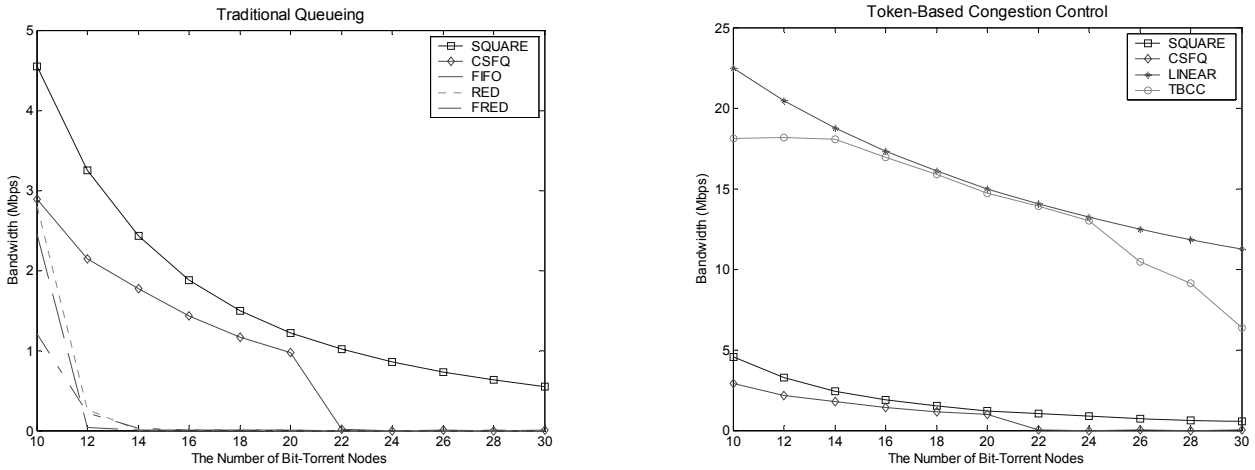
(a) (b)

Figure 6: Simulator results of traditional UDP applications competing with a P2P application based on UDP



(a) (b)

Figure 7: Simulator results for traditional TCP applications competing with a P2P application based on TCP



(a) (b)

Figure 8: Simulator results for traditional TCP applications competing with a P2P application based on UDP

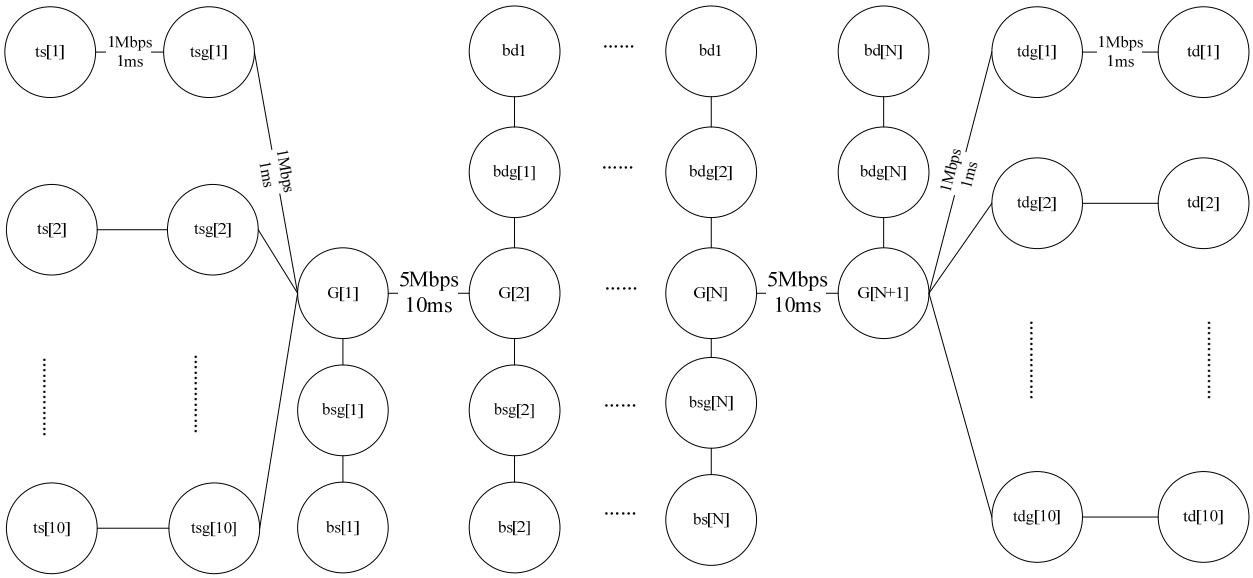


Figure 9: The topology for multiple congestion links in TBCC

not be fully utilized by the TCP flows. The second is the number large than 24. As the congestion degree can not increase enough high with the oscillation of the TCP flows, the P2P application can grab more bandwidth than it should.

The third simulator of single congestion link is that the traditional TCP applications just like FTP, compete with a P2P application of 2N nodes based on unresponsive UDP. In Figure 8(a), the queueings of FIFO, RED and FRED can not provide any protection to TCP flows. When the number of P2P nodes is large than 20, the access link become to full speed, the CSFQ also fail to protect TCP flows against the impact of UDP flows. In Figure 8(b), the character of TCP flows is the same as Figure 7(b), which demonstrates the TBCC can provide good protection to TCP flows against the impact of UDP flows by drop extra packets at edge router.

Multiple Congestion Links

We now analyze how the throughput of the traditional applications is affected when the flows traverse N congested links, where the N ranges from 10 to 30, the topology is shown in Figure 9. There are N congestion links from G[i] to G[i+1], where i ranges from 1 to N, the link bandwidth is 5Mbps, the link delay is 10ms, the ratio of high priority bandwidth is 90 %. The subscriber interior links between intelligent terminals and edge routers are scheduled by SPJBQ playing as a token bucket shaper, the enlarger is 10, the bandwidth is 1Mbps, and the link delay is 1ms..

There are ten traditional network applications competing with a P2P application of 2N nodes. The ten traditional network applications send packets from source ts[k] to destination td[k], where k ranges from 1 to 10. The P2P application sends packets from bs[i] to bd[j] while $i \le j$, and from bd[i] to bs[j] while $i < j$, all the flows traversing the core routers are from left to right, the total number of the flows is N^2.

We performed three experiments based on the topology shown in Figure 9, the simulation results are shown in Figure 10. The first is ten TCP applications competing with a P2P application based on TCP, the curve T2MT in Figure 10 shows its throughputs. The second is ten TCP applications competing with a P2P application based on unresponsive UDP, the curve T2MU in Figure 10 shows its result. The third is ten unresponsive UDP applications competing with a P2P application based on unresponsive UDP, the curve U2MU in Figure 10 shows its result.

According to the property of TBCC, we can deduce the throughput of traditional applications should decline linearly to the increase of the P2P Nodes, which is the curve LINEAR in Figure 10. All the curves show the same trend as the curve LINEAR, while the curve T2MU and T2MT have some random bias. According to the simulation results, It is evidence that the TBCC can provide fairness among subscribers even in multiple congestion links.

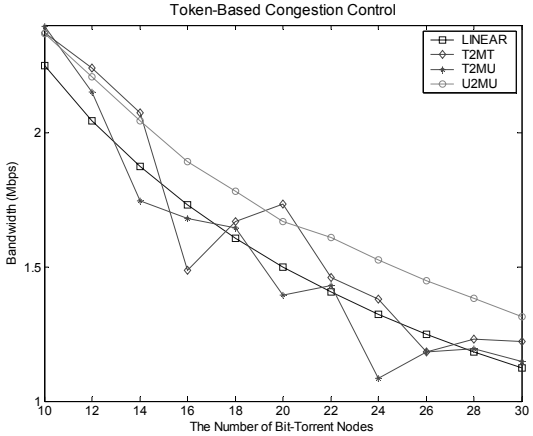


Figure 10: Simulator results for 10 traditional applications competing with a P2P application of 2N nodes in TBCC, where N ranges from 10 to 30.

Summarizing all the simulation results above, the TBCC can protect TCP flows better than the other four queueings, the TCP congestion control should be improved in TBCC to resist the impact of UDP flows.

4. Conclusions

This paper presents a new congestion control architecture TBCC, which can provide reasonably fair bandwidth allocation in proportion to the access token resource of each subscriber, and make the traffic of network more even. TBCC can also provide good protection to TCP flows comparison with CSFQ, but it is still expected to improve the TCP congestion control for achieving better throughput against the impact of UDP flows.

REFERENCES

[1] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent- like peer-to-peer networks. In Proc. of SIGCOMM, 2004.
[2] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance, ACM/IEEE Transactions on Networking, August 1993.
[3] Ion Stoica, Scott Shenker, Hui Zhang, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks", In Proc. of SIGCOMM, 1998.
[4] Abhay K. Patekh, "A Generalized Processor Sharing Approach Flow Control in Integrated Services Networks: The Single-Node Case", IEEE/ACM Trans. on Network, Vol. 1, No.3, June 1993.
[5] D. Lin & R. Morris, Dynamics of Random Early Detection, In Proc. of SIGCOMM, 1997.
[6] M. Nabeshima, Adaptive CSFQ: A New Fair Queuing Mechanism for SCORE Networks, IEEE HPSR, 2002.
[7] I. Stoica, H. Zhang, S. Shenker， Self-Verifying CSFQ, in Proceedings of INFOCOM, 2002.
[8] Floyd, S., Connections with Multiple Congested Gateways in Packet- Switched Networks Part 1: One-way Traffic. Computer Communications Review, Vol.21, No.5, October 1991, p. 30-47.
[9] Mahdavij, Floyd S, TCP-friendly unicast rate-based flow control, http://www.psc.edu/networking/papers/tcp_friendly.html#Flo 91, January, 1997.
[10] http://www.isi.edu/nsnam/ns/
[11] Andrew S. Tanenbaum, Computer Networks, Prentice-Hall International, Inc.
[12] Zhiqiang Shi, Jin Liang, Zhimei Wu, SPJBQ : Start Potential-based Jitter Bounded Queueing, IEEE SoftCom 2001.