Simulation and Exploration of RCP in the networks

Changhua He
changhua@stanford.edu
June 06, 2003

Abstract

RCP (Rate Control Protocol) is a rate-based congestion control algorithm, which can reduce the flow duration for small size files comparing with TCP slow start mechanism. Before transmission the sender and the routers will interact to achieve an optimal rate, in order to minimize the flow duration. We simulate and explore RCP in the networks with M/Pareto traffic. The result shows the optimal rate depends on the background traffic load and incoming traffic load of the bottleneck link. Furthermore, RCP works very well because some error in the optimal rate won't influence the flow duration greatly. Theoretically we work out the expression for queue size of a single queue with M/General traffic when the transmission rate is greater than the link capacity, and we describe a method to calculate the upper bound of the queue size when the transmission rate is less than the link capacity. Based on this, we explain the simulation results when M/General is specified as M/Pareto process.

1. Introduction

Congestion Control plays a very important rule in the current Internet to provide better performance. Nowadays the network capacity has increased significantly due to the implementation of optical links, but the amount of users and applications also increases greatly. Thus congestion control schemes are still necessary to limit flow rates to avoid "congestion" in the routers, to use the network resources efficiently to minimize flow durations, and to ensure "fairness" in resource allocation among flows.

The existing congestion control algorithms are mostly feedback schemes based on TCP [Wid01, Flyod01]. These schemes are window-based and the basis lies in Additive Increase Multiplicative Decrease (AIMD) [Chiu87, Jaco88]. The key principle of AIMD is halving the congestion window for every window containing a packet loss, i.e. congestion indication in wired networks, and increasing the congestion window by roughly one segment per RTT otherwise. The second fundamental component of TCP congestion control is the Retransmit Timer, including the exponential backoff of the retransmit timer when a retransmitted packet is itself dropped. The third fundamental component is the Slow-Start mechanism for the initial probing for available bandwidth, instead of initially sending at a high rate that might not be supported by the network. Within this general congestion control framework of Slow-Start, AIMD, and Retransmit Timers, there is a wide range of possible behavior dynamics. Furthermore, some researchers have tried to develop some congestion control schemes based on flow rate to provide TCP friendly behavior for multimedia traffic [Jain89, Sisa98, Reja99]. The common objective of these schemes is to find a suitable way for controlling traffic load according to the state of the network. Also some strategies are developed to constrain the traffic by using some formulas [Ban01], or based on the receiver-based mechanism [Hand02], so called TCP Friendly Rate Control (TFRC). This is a receiver-based mechanism, with the calculation of the congestion control information (i.e., the loss event rate) in the data receiver rather in the data sender. This is well-suited to an application where the sender is a large server handling many concurrent connections, and the receiver has more memory and CPU cycles available for computation. In addition, the receiver-based mechanism is more suitable as a building block for multicast congestion control.

However, the investigated protocols above are based on TCP, actually they will cause inefficiency when transferring small size flows. Fraleigh has measured the Sprint network and found the average flow size is less than 10,000 bytes and on average 10-15 packets (including SYN packet). With a Round Trip Time (RTT) of 100ms and the link capacity greater than 100,000 bytes/sec, which can almost always be satisfied in the current networks, the flow can be finished in only one RTT given the appropriate rate. While using TCP it will last about 500ms due to Slow Start and the average rate will be only 20Kb/s [Fral02]. This means TCP will “stretch” flows in such case. Hence, another protocol, RCP (Rate Control Protocol) is necessary in order to provide better performance. In this protocol, proposed by Rui Zhang-Shen and Nandita Dukkipati, the flow rate will be determined during the handshake process before data transmission by interaction between routers and the end-host, instead of by TCP probing. Then the flow will transmit packets with the given rate directly and thus reduce the flow duration. Using this scheme, each router tries to find the optimal rate to minimize the average flow duration for the active flows according to the traffic situation of itself.

2. Rate Control Protocol (RCP)

Intuitively, when a router assign high rates to the incoming flows, the flows can be finished in a short time, but might stay in the queue for a long time because of the increasing queue lengths caused by the high-rate flows. On the other hand, when a router assign lower rates to the flows, the average queue length will decrease, causing the lower packet delay, but the time to finish the flow will increase due to the low transmission rate. This is a kind of tradeoff. The basic idea of RCP is that, the router will assign some amount of capacity to the active flows based on the traffic situation of the network, minimizing the average flow duration.

Before a flow starts to transmit data, it first initializes a handshake process to determine the transmission rate. Each route receives the request will grant some amount of capacity according to its traffic situation and forward the handshake message to the next router along the route. The receiver will send an acknowlegement back to the sender indicating the suitable rate for all the routers. Then the sender will start transmission with the given rate directly. This is different from TCP, after handshake TCP will start transmission with one packet first and increase the rate by Slow Start, which actually “stretch” the small size flows. The handshake is drawn in Figure 1.

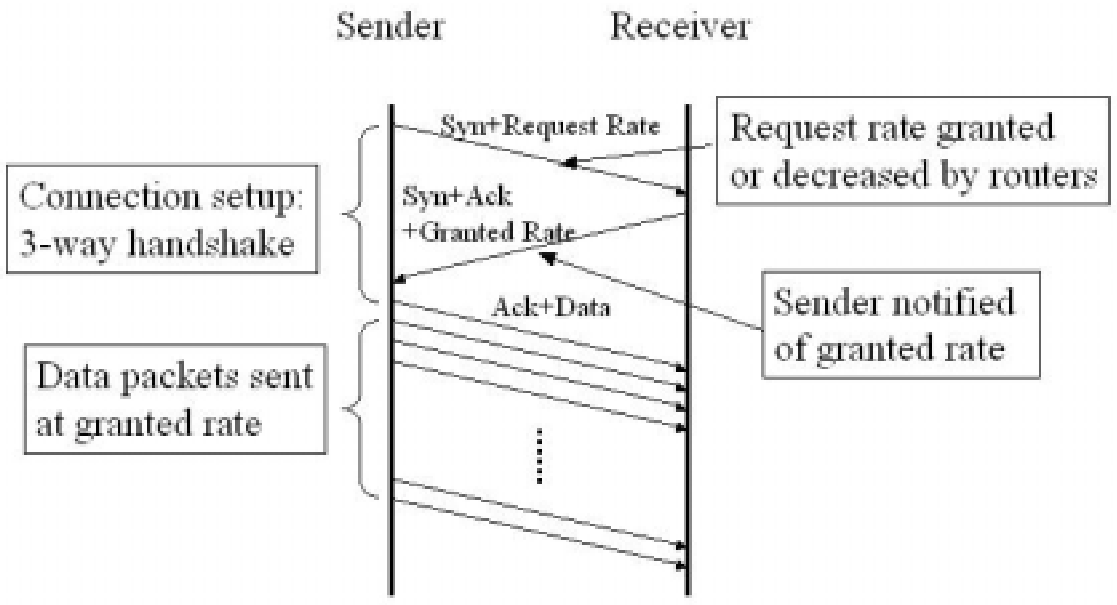
Sender
Receiver
Syn+Request Rate
Request rate granted or decreased by routers
Connection setup: 3-way handshake
Syn+Ack +Granted Rate
Ack+Data
Sender notified of granted rate
Data packets sent at granted rate

Figure 1. Hand-shaking process of RCP

3. Problem Statement

We can consider each router in the networks as a queue. Flow *i* has some size L_i. Each router *j* along the route from the sender to the receiver will grant rate R_j to the flow, after the handshake process the flow will use the minimum rate $R_k = \min\{R_1, R_2, \cdots, R_n\}$ to transmit data. The router *k* which assigns the minimum rate to the flow is called the bottleneck link for this flow.

The flow duration is defined as the time duration from the flow sends the first packet to the receiver receives the last packet, say, $T_{R,i} = TR_{R,i} - TS_{R,i}$, where $T_{R,i}$ represents the duration of flow *i* in the networks, $TR_{R,i}$ and $TS_{R,i}$ represent the start time of the first packet and the received time of the last packet, respectively. Easily we have the relationship $T_{R,i} = L_i / R_k + D_{R,i}$, where $D_{R,i}$ is the packet delay, which is the delay time of the last packet, and the first item L_i / R_k is the transmitting time of the flow.

In each router, it will assign the same rate to all the active flows. The objective of RCP is to determine the rate to minimize the average flow duration. Once the optimal rate is determined, the router will grant the rate to the flows and it tries to operate at this optimal point.

In a single link case, the router will determine the rate and the operation point thus determine the average flow duration. We only need to explore how the background traffic will influence the optimal operation point. While in the network with many routers along the path of the flow, each router will assign rates according to its own traffic situation, which is more complex because the diversity of the router capacities and the traffic situations. This case can be considered as several queues in series along the route of the flows as shown in Figure 2.

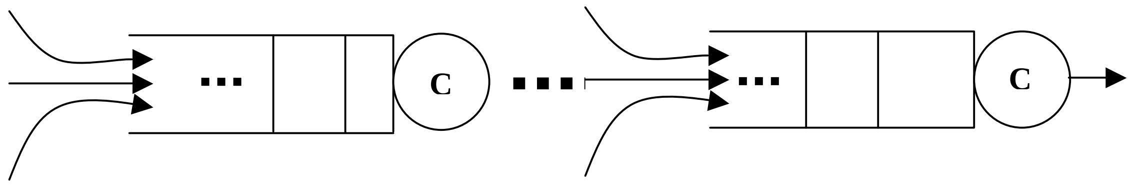

Figure 2. Queue Models during the flow transmission

The objective of this project is to explore RCP dynamics over the networks, hence find out how well RCP works, how the flow duration relates to the traffic parameters, and how to determine the optimal rate for transmission to minimize the flow duration.

4. Simulation

First we will run enough simulations to get some intuition on the algorithm and the performance. In the real networks, background traffic exists for every router and obviously it will affect the behavior of the router. Thus in the simulation we will consider the traffic in each router consists of background flows and interested flows. The queues along the route of the interested flows are our studying object. For the background flows in each queue, the relationship is quite complicate based on the topology and traffic pattern, here for simplicity we will assume the background flows in different queues are independent.

In the current Internet, the traffic has been proved to be heavy-tailed (Long Range Dependence) [Lela94, Paxs95, Feld98], and M/Pareto process is suitable to model and simulate such kind of traffic [Neam99]. Hence we will use this traffic model to generate both background flows and interested flows, say, the flow arrival is Poisson process with arrival rate λ, the flow size L is Pareto distributed and each flow has a transmission rate R. We set the capacity of each router to C, and easily the average load of the router is,

$$\rho = \frac{\lambda EL}{C}$$

In the following simulations, we always assume admissible traffic for each router, say, $\rho < 1$.
Among all the queues along the route of the interested flows, the one that assigns the minimal rate to the flows is considered as the bottleneck link for these flows. Because the router will assign the same rate to all the flows passing it, we can easily conclude in the bottleneck link, the background flows will have a rate no greater than the interested flows, otherwise there must be another link that assigns a lower rate to the interested flows, and become the bottleneck link.

Rui and Nandita have obtained some results on the single link case and developed RCP algorithm based on the observation on the results. However, the background traffic might change the behavior of the router, therefore at the first step we will simulate the single link case with background flows. Throughout the simulations, we will keep the capacity of the link as 100Mbps, delay of the link as 100ms, the Pareto shape as 1.1, the mean as 10 packets and each packet has a size of 1000 bytes.

Case1: Single Link with background traffic
The link has 20000 interested flows incoming, which cause an average load ρ_I, and each flow has a transmission rate r_I. 30000 background flows exists with an average load ρ_B, each flow should have a very low transmission rate, we set it to $r_B = 0.02$ and keep it unchanged. We can calculate ρ_I and ρ_B respectively from the flow information using above formula.
First, we consider the link has a relatively high load of background traffic, say, $\rho_B = 0.4972$, and change ρ_I, r_I, we get the average flow duration and queue length as shown in Figure 3. (the transmission rate in the figure is normalized by the link capacity.)

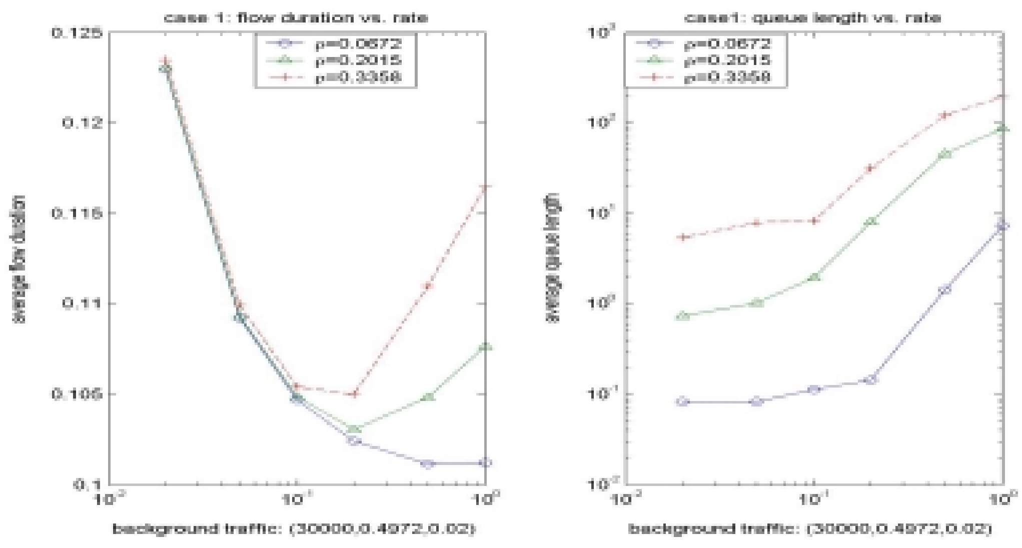

Figure 3. Flow duration and queue length vs. rate of interested flows (case 1: high load)

Intuitively when rate is low, the transmission delay is large and packet delay is small, while when rate is high, the transmission delay is small but packet delay is large. Consider this kind of tradeoff we should have some optimal transmission rate to minimize the flow duration. From Figure 3, we verify this is true with background traffic existing in the link. Also we can find around the optimal point the curve is rather flat, that means the optimal point is not very sensitive to the rate change. For example, for the red curve with $\rho_I = 0.3358$, there are no large variations in flow duration when the rate changes between 0.1 and 0.2. Furthermore, When ρ_I increases (from 0.0672 to 0.3358), the optimum point moves towards left side, which means the router should assign a lower rate to each flow when the load of interested flows increases.

Now we want to see how the background traffic will influence the flow duration. Generally the transmission rate of the background flow is very slow, so we will keep it at the previous value, but change ρ_B from high load (0.4972) to low load (0.1657) with all other parameters unchanged.

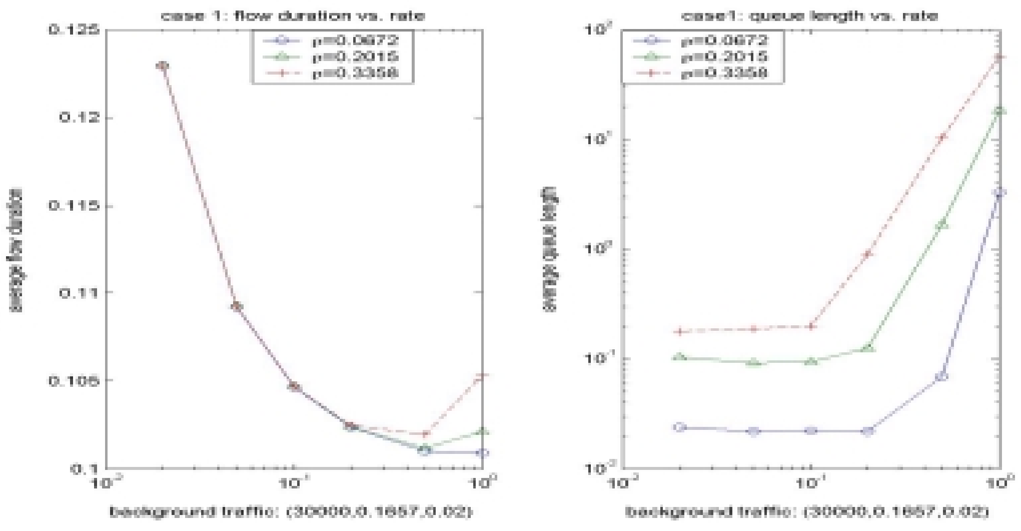

Figure 4. Flow duration and queue length vs. rate of interested flows (case 1: low load)

In Figure 4 the shape of the curve is the same as before, also we get the optimal rate with a rather flat part around it. But the difference is, for the same ρ_I, the optimal point appears at a higher rate. For example, for $\rho_I = 0.3358$, the optimal rate is around 0.2 in Figure 3, while it appears around 0.5 in Figure 4. That means, when the load of background traffic is lower, we should assign higher rate to each flow to reduce the transmission delay thus the flow duration because the packet delay won't take much effect.

After simulating a single link with background traffic, we will consider more complex cases, say, multiple links in series along the route of the interested flows.

Case2: Two Links(the second link is bottleneck)

In this case, we set the second link with $\rho_B = 0.4972$ and $r_B = 0.02$, which is the same as the high load situation in the single link case. We then add a non-bottleneck link before it. For the added link, there are also 30000 flows. But the average load is lower, and we are interested in the

situation that the background traffic has a relatively high transmission rate, hence it is possible to influence the flow duration much. We set them to $\rho_{B1} = 0.1471$, $r_{B1} = 0.2$.
We adopt the same interested flow sets, fed into the non-bottleneck link first then the bottleneck link. The results are shown in Figure 5.

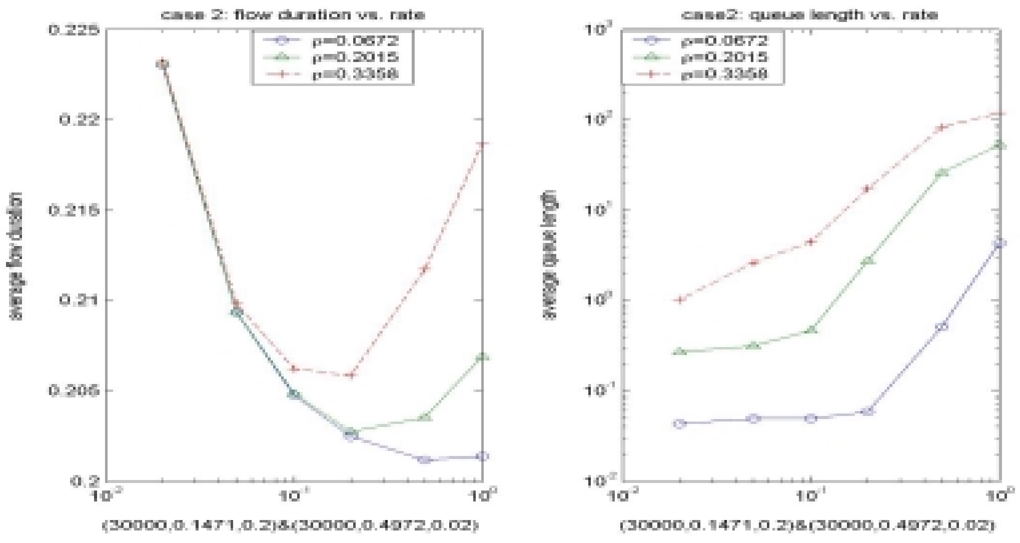

Figure 5. Flow duration and queue length vs. rate of interested flows (case 2)

Comparing with Case 1, the curve is almost a shift version of that in Figure 3 towards upside. That means, the first link we added only brings a constant delay to the flows and does not change the shape of the curve, again we have a rather flat part around the optimal rate. From these we can see the system is scalable and the optimal rate of minimizing flow duration is determined by the bottleneck link. We do not need to add more non-bottleneck link and simulate because they will just add a constant delay similarly.

In case 2, the interested flows are fed to the first non-bottleneck link, after shaped by this link they enter the bottleneck link. We are still interested in whether things will change if the interested flows are fed to the bottleneck link directly. So we setup case 3 to test whether the relative position of the bottleneck link will have any influence on the result.

Case3: Two Links(the first link is bottleneck)
In this case, we will keep everything the same as in case 2 except exchanging the position of the two links. Currently the interested flows are fed into the bottleneck link directly and then the non-bottleneck link. The results are shown in Figure 6. Comparing with Figure 5, we can see there are only minor differences between them when the load of interested flows are high (0.3358), and no differences when the load of interested flows are low. That is because with the high load of interested flows fed into the bottleneck link directly, the total load will be rather high, the queue increases to cause a high packet delay, the bottleneck link should assign lower rate to the incoming flows. But generally when the total load is not so high, the flow duration and optimal point is not very sensitive to the relative positions of the bottleneck link.

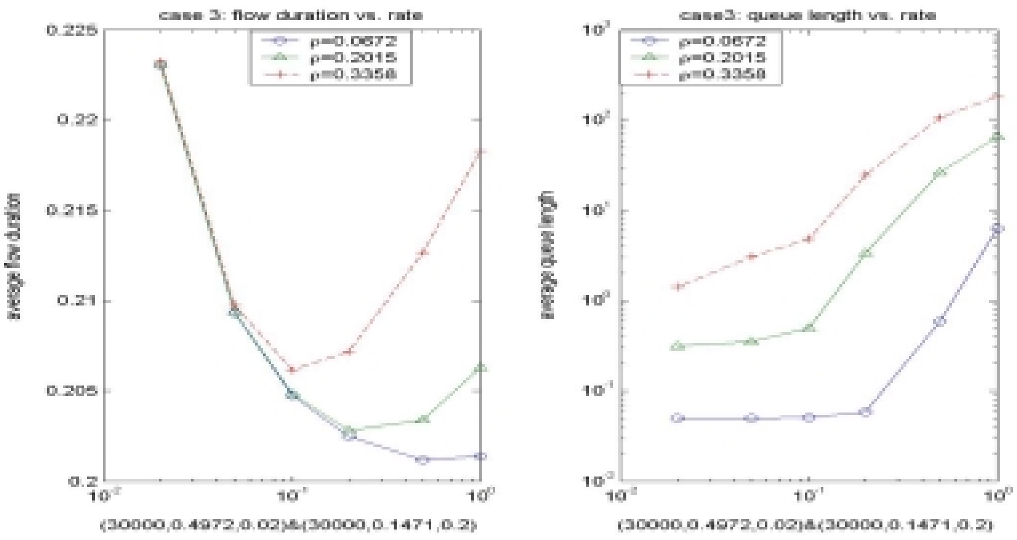

Figure 6. Flow duration and queue length vs. rate of interested flows (case 3)

Summary of observations:

Based on above simulations, we have the following findings:
(1) For heavy-tailed traffic, which is true in the real networks, the optimal rate for flow transmission exists, and around the optimal value the curve is rather flat, which means we have no strict requirement when assigning the rate.
(2) Each router can determine its optimal rate depends on both the background traffic load and the incoming traffic load. The higher the load, the lower the optimal rate.
(3) RCP can work very well with series of links in the networks because it is scalable and the optimal rate is determined only by the traffic situation of the bottleneck link.

In practice, each router will decide the rate depending on the traffic situation of itself, say, the load of background (ongoing) flows and the load of interested (incoming) flows. Fortunately these two parameters can be observed easily by the router. In the simplest way, the average queue length will indicate the background traffic load and the queue increasing rate indicates the incoming traffic load. Hence each router can assign the suitable rate to the flows periodically based on their observations on the traffic.

5. Analysis & Modeling

From the simulation we have achieved some useful results, it will be a better guidance if we can do some theoretical analysis. As discussed in Section 2, we can divide the flow duration to the transmission delay and the packet delay. It is very difficult to calculate the packet delay, so we will focus on the queue length and approximate the packet delay by the queue length, although we don’t really know the relationship between them. For simplicity in the analysis we only consider the single link without background traffic, drawn in Figure 7.

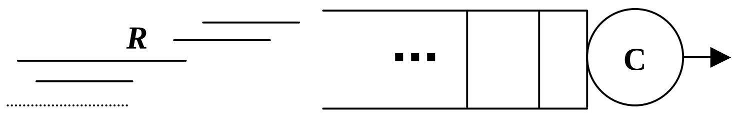

Figure 7. Queue model of single link without background traffic

Assume the queue length can be infinity, let's rewrite the problem as follows:
The link capacity is C, and the server will serve packets in the queue in a ***packet-based FCFS*** way, say, the first arrived packet will be served first. Flows arrive according to a Poisson process with arrival rate λ, the flow size L is any general distribution and each flow has a transmission rate R, our objective is to calculate the average queue length (in bytes).

When the flow size L is exponentially distributed, Pan et al [Pan91] and Kosten [Kost74] have solved the problem. But that is not the case in real networks. Here we will consider the flow size is any general heavy-tailed distribution. Because we can't get a general close-form expression for the queue length in such situation, instead we will try to analyze some special values of R, $R=\infty$, $R=C$, $R=\frac{C}{n}$, and so on, and try to verify the shape of the curve.

5.1 $R=\infty$
When the transmission rate is infinity, each flow will arrive as a whole at a time point, this is exactly an M/G/1 queue system. We have following results from Pollaczek-Khinchine formula:

$$EW_\infty = \frac{\lambda ES^2}{2(1-\lambda ES)}$$

where ET_∞ is the average flow duration, EW_∞ is the waiting time, ES_∞ and $ES_\infty{}^2$ are the first and second moment of the flow service time.

Because the flows arrive according to Poisson process, from PASTA, the time average queue size is the same as the flow arrival seen average, we have

$$EQ_\infty = C \cdot EW_\infty = C \cdot \frac{\lambda ES^2}{2(1-\lambda ES)} = \frac{\lambda C}{2(1-\rho)} ES^2$$

where EQ_∞ is the time average queue size (in bytes).

5.2 $R=C$
In this case, instead we consider the following ***flow-based FCFS*** Model:
The flow arrivals are exactly the same as in Figure 7, but the service discipline is different. In this system the service is flow-based FCFS, instead of packet-based FCFS in our system. That means, when the first packet of a flow is served, the server will continue to serve all the packets of the same flow until it is completed, no matter packets of other flows arrive or not.

Let EQ_R be the queue length in our system, EQ_F be the queue length in FCFS system. We know these two system will have the same queue length, say, $EQ_R = EQ_F$, because at any time, the arrivals are the same and the amount of packets served (no matter which flow the packets come from) are also the same.

On the other hand, comparing the flow-based FCFS model with M/G/1 queue, we can conclude the departures are exactly the same (packet by packet), thus the only reason that causes the different queue size is the arrivals. Generally we have

$$EQ_R = \lim_{t\to\infty} \frac{\int_0^t [A_R(t) - D_R(t)]dt}{t}$$

where $A_R(t)$ represents the total size of packets arrived from *0* to *t*, $D_R(t)$ is the total size of packets leaves the queue from *0* to *t*, assume at *t=0* the queue is empty.

We can draw the arriving process in the M/G/1 queue and the flow-based FCFS system as in Figure 8. The dark arrow represents the arrival of M/G/1 queue and the light line represents the arrival of flow-based FCFS system.

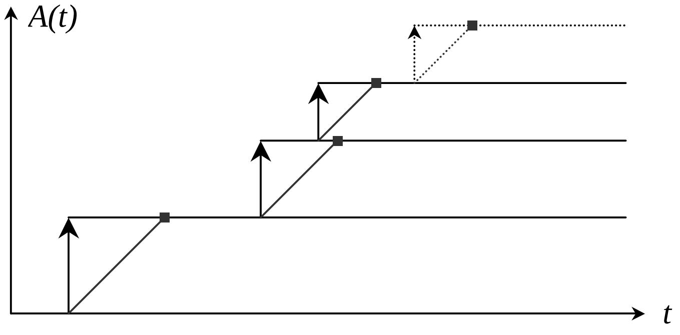

Figure 8. Arrival process of M/G/1 queue and flow-based FCFS system

The queue difference of these two systems can be determined by the difference of the arrival process, say, the area of the triangles in Figure 8. Assume from *0* to *t* total $N(t)$ flows arrive,

$$\Delta Q_R = \lim_{t\to\infty}\frac{1}{t}\left[\int_0^t (A_\infty(t) - A_R(t))dt\right] = \lim_{t\to\infty}\frac{1}{t}\sum_{i=1}^{N(t)}\frac{1}{2}\frac{L_i^2}{C} = \lim_{t\to\infty}\frac{N(t)}{t}\frac{1}{N(t)}\sum_{i=1}^{N(t)}\frac{1}{2}\frac{L_i^2}{C} = \lambda\frac{EL^2}{2C} = \frac{\lambda C}{2}ES^2$$

so $EQ_R = EQ_F = EQ_\infty - \Delta Q_R = \dfrac{\lambda C}{2(1-\rho)}ES^2 - \dfrac{\lambda C}{2}ES^2 = \dfrac{\rho\lambda C}{2(1-\rho)}ES^2$

5.3 $R > C$

Using the same method as in Section 5.2, we have

$$\Delta Q_R = \lim_{t\to\infty}\frac{1}{t}\sum_{i=1}^{N(t)}\frac{1}{2R}L_i^2 = \lambda\frac{EL^2}{2R} = \frac{\lambda C^2}{2R}ES^2$$

$$EQ_R = EQ_F = EQ_\infty - \Delta Q_R = \frac{\lambda C}{2(1-\rho)}ES^2 - \frac{\lambda C^2}{2R}ES^2 = \frac{\lambda C}{2}\left(\frac{1}{1-\rho} - \frac{C}{R}\right)ES^2$$

From the result we can see when the flow size is Pareto distribution (heavy-tailed) with the shape between 1 and 2, the queue size is infinity when $R \ge C$ because $ES^2 = \infty$, which drives the packet delay to infinity.

5.4 $R < C$

We just consider the special points where $R = \dfrac{C}{n}$, $n \ge 1$ is an integer. Similarly, we can construct another system with *n* parallel flow-based FCFS queues, each queue has a capacity of $\dfrac{C}{n}$, drawn as in Figure 9. Assume the scheduler is intelligent and knows the size of each flow, it will always allocate the incoming flows to the sub-queue with the least unfinished work (in bytes). Each sub-queue is flow-based FCFS as we discussed in Section 5.2. We will account the queue size of this system as the sum of the queue size of all sub-queues.

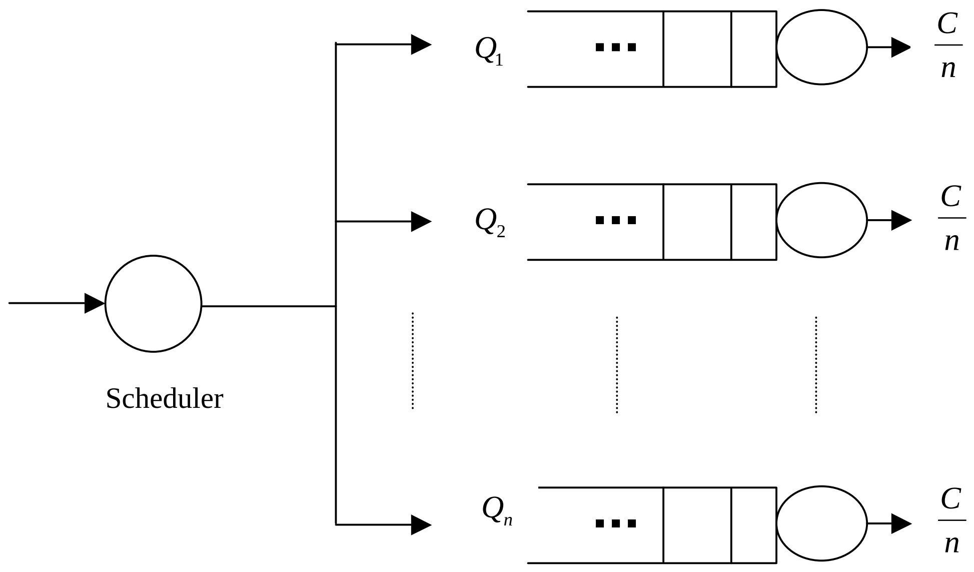

Figure 9. *n* parallel flow-based FCFS queues

First of all, unlike the analogy in Section 5.2, the queue size of the system shown in Figure 9 is not exactly the same as in our system. Whenever there are queues built-up (arrivals greater than service), in our system the queue will be drilled by all available servers later, while in the *n* parallel queues the built-up queue distributes in the separate sub-queues and can be drilled out by the only server of the sub-queue when it is free. However, this queue size is an upper bound of the queue size of our system, because the arrivals are the same and the accumulative departures are less in this system. This kind of upper bound is also useful in practice.

In order to calculate this upper bound, similarly we compare it to an M/G/n queue. The departures are the same as an M/G/n queue. Thus we can do the same thing as in Section 5.2 to calculate the average queue size. However, we can't expect to get the close-form expression here because until now there are no direct close-form results for an M/G/n queue. We might use some other methods to approximate or analyze this kind of queue, which we won't discuss here.

5.5 $R \to 0$

Here we will use the system drawn in Figure 9 to show $EQ_R \to 0$ as $R \to 0$.

Instead of the "intelligent" scheduler which allocates incoming flows to the sub-queue with Least Unfinished Work First policy, consider a "stupid" scheduler that just allocates the incoming flows to the sub-queue independently and randomly, obviously we have $EQ_R \le EQ_{F,LWF} \le EQ_{F,Rand}$, where EQ_R is the queue size of our system, $EQ_{F,LWF}$ is the queue size of the system with "intelligent" scheduler, and $EQ_{F,Rand}$ is the queue size of the system with "stupid" scheduler.

For each sub-queue in "stupid" scheduler system, its arrival process is Poisson with rate $\frac{\lambda}{n}$, so

$$
\begin{aligned}
&\Pr(there.is.queue.built.up.in.this.subqueue.at.time.t)\\
&= \Pr(arrival.rate.greater.than.service.rate.at.time.t)\\
&= \Pr(first.flow.continus.at.any.time.t.\&.one.or.more.flows.arrive.before.t)\\
&= \Pr\left(\frac{L}{C/n} > t\right)\cdot\{1-\Pr(no.flows.arrive.before.t)\}\\
&= \Pr\left(L > \frac{C}{n}t\right)\cdot\left(1-e^{-\frac{\lambda}{n}t}\right) \to 0 \quad as \quad n\to\infty, R\to 0
\end{aligned}
$$

This follows $EQ_{F,Rand} \to 0 \quad as \quad R \to 0$.

$\therefore \quad EQ_R \to 0 \quad as \quad R \to 0$

5.6 Explanation

Based on above analysis, we have shown when $R \to 0$ the queue size goes to zero thus the packet delay goes to zero approximately. As in the real networks, when the flow size is Pareto distribution (heavy-tailed) with the shape between 1 and 2, the queue size is infinity when $R \ge C$ due to $ES^2 = \infty$, thus the packet delay is infinity.

On the other hand, for the transmission delay, it decreases monotonically from infinity to zero when *R* increases from zero to infinity. Hence when we add them up to get the flow duration, we will obtain the curve shape as shown in our simulations (though we do not prove it in rigor) and the optimal rate exists somewhere between 0 and R. However, in the simulation result, when $R = C$ the queue size and flow duration is not big enough to be considered as infinity, that is because the simulation time is finite and the maximal flow size is limited by the computer, we can't generate the real Pareto distribution with infinite variance in a short running time.

6. Conclusion

In this project, we have done the simulations and based on the data, we found RCP can work very well in the networks because the optimal rate will minimize the flow duration and the flow duration does not change much will some error on the optimal value (the curve is flat around the optimal rate). The optimal rate is determined by the bottleneck link. It is scalable and not sensitive to the relative positions of the bottleneck link. For each router, it can determine the rate only depending on the traffic situation of itself, say, the load of background (ongoing) flows and the load of interested (incoming) flows.

Furthermore, we try theoretical analysis for the single queue without background traffic. We achieve the close-form expression for the queue size when $R \ge C$, describe the method to calculate the upper bound of the queue size when $R < C$, and prove that $EQ_R \to 0$ as $R \to 0$. Also from the results, when the flow size is Pareto distributed with the shape between 1 and 2, we have a rough explanation to show the curve of flow duration vs. rate should shape as shown in our simulations and the optimal rate exists somewhere between 0 and R .

7. Future Work

Until now, we have achieved fairly good results in both simulations and theoretical analysis. However, there are still lots of work left. One of them is, in the analysis, we only consider the queue size of the system, and approximate the packet delay by the queue size. But we don't know how does the packet delay relate to queue size exactly. Thus that will be very interesting to do some researches on the packet delay. Furthermore, we can't get a close-form expression for the queue size when $R < C$ because nobody solve the M/G/n problem, but it is valuable to analyze how flat the curve is around the optimal rate, that can show us how much error is tolerable when a router assign the rate to the flows.

8. Acknowlegement

Special acknowledgements go to Rui Zhang-Shen and Nandita Dukkipati for continuous assistance on the project. We are also grateful to Isaac Keslassy for offering kind help on discussing and handling the project. Finally, acknowledgement goes to Prof. Nick Mckeown for useful comments on the scope and depth of the project.

References:

[Ban01] D. Bansal, H. Balakrishnan, "*Binomial Congestion Control Algorithms*", IEEE INFOCOM 2001, Apr. 2001.

[Chiu87] D.M.Chiu, R.Jain, "*Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks*", DEC-TR-509, Aug. 1987.

[Feld98] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz, "*The Changing Nature of Network Traffic: Scaling Phenomena*", ACM Computer Communication Review, vol. 28, pp. 5-29, Apr. 1998.

[Floy01] S. Floyd, "*A report on recent developments in TCP congestion control*", IEEE Communications Magazine, pp. 84-90, April 2001.

[Fral02] Charles J. Fraleigh, "*Provisioning Internet Backbone Networks To Support Latency Sensitive Applications*", PhD dissertation, May 2002.

[Hand02] M. Handley, J. Padhye, S. Floyd, J. Widmer, "*TCP Friendly Rate Control(TFRC): Protocol Specification*", draft-ietf-tsvwg-tfrc-03.ps, July 2001, exp. Jan. 2002.

[Jaco88] V.Jaconson, "*Congestion Avoidance and Control*", ACM SIGCOMM'88, p314-329.

[Jain89] R.Jain, "*A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks*", DEC-TR-566, Apr. 1989.

[Kost74] L. Kosten, "Stochastic Theory of a Multi-entry Buffer (I)", Delft Progr. Rep., Series F: Mathematical Engineering, Mathematics and Information Engineering, pp. 10-18, 1974.

[Lela94]]Will Leland, Murad Taqqu, Walter Willinger, and Daniel Wilson, "*On the Self-Similar Nature of Ethernet Traffic (Extended Version)*", IEEE/ACM Transactions on Networking, Vol. 2, No. 1, pp. 1-15, February 1994.

[Neam99] T. D. Neame, M. Zukerman and R. G. Addie, "*Application of the M/Pareto Process to Modeling BroadBand Traffic Streams*", Proc. of ICON'99, pp53-58, Brisbane, Queensland, Australia, 28 September, 1999.

[Pan91] Huanxu Pan, Hiroyuki Okazaki and Issei Kino, "*Analysis of a Gradual Input Model for Bursty Traffic in ATM*", TELETRAFFIC AND DATATRAFFIC, 1991.

[Paxs95] V. Paxson and S. Floyd, "*Wide-area traffic: The failure of Poisson modeling*", IEEE/ACM Trans. On Networking, 3(3):226-244, June 1995.

[Reja99] R.Rejaie, M. Handley, D. Estrin, "*RAP: An end-to-end rate based congestion control mechanism for real-time streams in the Internet*", in Proc. Of IEEE INFOCOM'99, Vol.3, pp.1337-1345, Mar. 1999.

[Sisa98] D.Sisalem, H. Schulzrinne, "*The Loss-based Adjustment Algorithm: A TCP-friendly Adaptation Schemes*", in Proc. Of NOSSDAV'98, Cambridge, England, July 1998.

[Wid01] J. Widmer, R.Denda, M.Mauve, "*A survey on TCP-friendly congestion control*", IEEE Network, pp. 28-37, May/June 2001.