CARPATHIAN J. MATH.
20 (2004), No. 2, 149 - 154

Improving the Round Trip Time Estimation in Internet Routers

DARIUS BUFNEA, ADRIAN STERCA, CLAUDIU COBÂRZAN, FLORIAN BOIAN

ABSTRACT. We review in this paper the concept of TCP-friendliness and we debate one of its weaknesses. Unfortunately, this weakness reduces the practical applicability of the TCP-friendliness equation. For a very high percent of the Internet flows, the TCP-friendliness test may have wrong results. We suggest in this paper a router algorithm that improves the efficiency of the TCP-friendliness equation for some UDP flows. This algorithm uses the additional information retrieved by the router from competing TCP flows in order to obtain a better approximation for the round trip time as used in the TCP-friendliness test.

1. INTRODUCTION

The last decade has confirmed the expected growth rate of the Internet community. Nowadays, this continuing evolving community is having new needs regarding running Internet applications. New protocols have emerged to support these new applications, mostly audio- and video-based. Also, these applications have lead to an increase in the amount of traffic flowing in the current Internet. Unresponsive or bad designed protocols, non-specialized users and the increased Internet traffic are the main causes responsible for the appearance of an unwanted phenomenon, known as congestion collapse.

Congestion occurs in a computer network when the number of packets delivered to destination is significantly smaller than the number of packets injected into the network. This situation is very common when the network load increases over the links' capacities, i.e. when the number of packets injected into the network reaches or surpasses the network transport capacity. Routers in the middle of the path from source to destination are constrained to discard a high number of queued packets. These packets, before being discarded, had consumed precious network resources such as link bandwidth, router memory and processor time. More over, discarding packets leads to unwanted behavior for user applications. For an overview about congestion control roots, preventing and treatment measurements, the reader should consult [4] and [7]. Congestion control algorithms fall into two classes, depending on where they are used. Router based algorithms and techniques for congestion control [3] try to detect congestion in its incipient phase, i.e. preventing it, or in the worst case, resolving the congestion situation by discarding or marking packets. All this work is done at the communication network level. Protocols based congestion control algorithms [1] try to resolve congestion situations by using implicit or explicit feedback received by the sender

Received: 26.09.2004; In revised form: 12.01.2005
2000 *Mathematics Subject Classification.* 60K30, 68M10, 68M12.
Key words and phrases. *congestion control, TCP friendly, TCP time stamp, UDP flows.*

from the communication network. These two classes of algorithms do not exclude each other. Rather, they are complementary.

Contributions. We propose a new router based algorithm that can be used to improve the detection process of non TCP-friendly UDP flows in some situations. Our improvements are based on the additional information collected by routers from TCP flows. We are especially interested in those TCP packets having the same source or destination address as an UDP flow suspected not to be TCP-friendly. Routers can use the time stamp information extracted from these packets [5] in order to improve the estimated round trip time value used in the TCP-friendliness equation. The round trip time value used in the TCP-friendliness equation, as proposed in [2], is far too weak. This drastically reduces the number of cases when the TCP-friendliness equation can be applied, or leads to wrong conclusions if applied in improper network conditions.

The paper is organized as follows. The next section reviews the main concepts used in this paper such as TCP-friendly flow and congestion aware protocol. It also debates the TCP-friendly test equation and some of the disadvantages of using it in certain situations. The algorithm for improving the round trip time estimation in Internet routers is revealed in section 3. The paper ends with the results of our experiments using the proposed algorithm, together with considerations of future work and improvements of our mechanism.

2. The TCP-friendliness test

The data in Internet is currently carried over two types of protocols. Congestion aware protocols, such as TCP, reduce their source output as congestion occurs. Congestion occurrence is reported to its source either directly using ECN [6] or indirectly by packet drops observation. As a response, the network load should decrease and the congestion situation resolved.

Congestion unaware protocols, such as UDP, are unresponsive to congestion. This behavior can lead to the unfairness problem caused by unresponsive flows that consume more bandwidth that a regular flow which obeys congestion control policies. In this situation, when a misbehaved flow gets a larger amount of bandwidth than a good behaved one, new network incentives are needed in order to prevent such a behavior.

In [2], Floyd and Fall introduced the concept of TCP-friendly flow and gave an equation serving to the identification of misbehaved UDP flows, i.e. not TCP-friendly.

Definition: A flow is TCP-friendly if its arrival rate does not exceed the bandwidth of a conformant competing TCP connection in the same network circumstances.

In order to determine if a flow is or not TCP-friendly, we can use the TCP-friendliness test given in [2]. This test is based on the following equation, which gives the maximum sending rate of a conformant TCP connection:

$$T \leq \frac{1.5 \times \sqrt{2/3} \times B}{R \times \sqrt{p}} \tag{2.1}$$

In the above equation, B is the packet size, p is the packet drop rate and R is the round trip time estimated for the conformant TCP connection. The arrival rate of a TCP-friendly flow should not be higher than T bytes/sec. Any flow, with an arrival rate T higher than $\frac{1.5\times\sqrt{2/3}\times B}{R\times\sqrt{p}}$ will not be TCP-friendly and the router should apply to this flow proper incentives, i.e. bandwidth restriction or packet drops.

This equation is directly derived from the TCP additive increase, multiplicative decrease (AIMD) congestion control algorithm [1]. The packet size B and the packet drop rate p of a flow can be easily determined through direct observation by the router. But, for estimating the round trip time value R, there is no simple method, as the authors of [2] have admitted. They suggest setting R to twice the one-way propagation delay of the attached link - i.e. the link where the packet is being sent through. This approach reduces the practical usage of this test to those routers having a significant propagation delay of the attached link in the end-to-end delay of a connection's path.

An average path length in the current Internet is around fifteen hops. The propagation delay of every link of this path is contributing twice to the end-to-end delay of a connection - once for a packet and once for its acknowledgement. If the outgoing attached link's delay is not a significant fraction of the end-to-end delay of a connection's path, the round trip time for this connection is underestimated by a factor of thirty. Improper usage of this value in a TCP-friendliness test could lead to an overestimated allowed sending rate. A misbehaved UDP flow can use in these circumstances, thirty times more bandwidth than a conformant TCP flow, without failing the TCP-friendliness test. For longer Internet paths, the allowed overestimated sending rate could be fifty times higher than the sending rate of a good behaved flow.

3. Improving the RTT approximation for some UDP flows using TCP time stamps

The TCP protocol uses a time-stamp option in order to compute a connection's round-trip time [5]. Unfortunately, a similar approach for an UDP flow is inappropriate as the UDP protocol is not connection oriented. In this situation, an application must rely on its own internal mechanism for computing the round trip time. This round trip time information, even if present, it is meaningless for a router process running at network or transport level.

We will use the following notations to present and explain our algorithm:

R - a router application or hardware implementing our algorithm or the Internet address of this router;

F - an UDP flow suspected of not being TCP-friendly;

$UDPaddr_s$ and $UDPaddr_d$ - the source and the destination address of flow F, as seen by router R;

P - a TCP packet forwarded by router R;

$TCPaddr_s$ and $TCPaddr_d$ - the source and the destination address of packet P, as seen by router R;

A_P - the acknowledgment of packet P. A_P will have $TCPaddr_d$ as the source address and $TCPaddr_s$ as the destination address;

The time stamp option ($TSValue$,$TSEchoReply$) - a TCP time stamp option enclosed in a TCP packet as specified in [5]. The $TSValue$ is meaningful for us only in packet P, while the $TSEchoReply$ is only meaningful in its acknowledgment A_P. Let (P_{TS}, x) be the time stamp enclosed in P, and let (y, P_{TS}) be the time stamp enclosed in A_P. The x and y element of the above tuples are meaningless for our algorithm.

Our algorithm tries to approximate the round-trip time on the $UDPaddr_s$-$UDPaddr_d$ path using at least one of the partially round-trip times $UDPaddr_s - R$ or $R - UDPaddr_d$. In order to achieve this, the router will scan for forwarded TCP packets having the destination address $TCPaddr_d \in \{UDPaddr_s, UDPaddr_d\}$. These packets may belong to competing well-behaved TCP connection. Let P be such a packet, having $TCPaddr_d = UDPaddr_d$. After processing packet P, the router stores in a hash table the local time, keyed by the $(TCPaddr_s, TCPaddr_d, P_{TS})$ tuple. When packet P is acknowledged, the router looks in the hash table after the $ObservedTime$ value referred by the $(TCPaddr_s, TCPaddr_d, P_{TS})$ key - all these tuple members can be retrieved from A_P. If such a value is found, a round trip time sample for the $R - UDPaddr_d$ path can be computed as $RouterLocalTime - ObservedTime$. We will use this computed value as an approximation for the round-trip time on the $UDPaddr_s - UDPaddr_d$ path. If round-trip times can be computed for both $UDPaddr_s - R$ and $R - UDPaddr_d$ paths, we will use their sum instead.

```
Algorithm ComputeUDPFlowRTTApproximation is:
Input:  A suspected misbehaved UDP flow $F$ specified by source address
   $UDPaddr_s$ and destination address $UDPaddr_d$.
Output:  The smooth round-trip time approximation $SRTT_{UDPFlow}$ for the
   UDP flow $F$.
Begin
   $InterestedIPs = \{UDPaddr_s, UDPaddr_d\}$;
   For a forwarded TCP packet $P$ having the source address $TCPaddr_s$,
      destination address $TCPaddr_d$ and the enclosed time-stamp option
      $(P_{TS}, x)$:
         If $TCPaddr_d \in InterestedIPs$ then
            $ObservedTime = RouterLocalTime$;
            HashTableEntry ($TCPaddr_s$,$TCPaddr_d$,$P_{TS}$) = $ObservedTime$;
         End If;
   End For;
   For a forwarded acknowledgement packet $A_P$ having the source address
      $TCPaddr_d$, destination address $TCPaddr_s$, the enclosed time-stamp
      option $(y, P_{TS})$:
         If $TCPaddr_d \in InterestedIPs$ then
            $ObservedTime$ = GetHashTableEntry ($TCPaddr_s$,$TCPaddr_d$,$P_{TS}$);
            RemoveHashTableEntry ($TCPaddr_s$, $TCPaddr_d$, $P_{TS}$);
            $MeasuredRTT = RouterLocalTime - ObservedTime$;
            Use $MeasuredRTT$ to compute the smooth round trip $SRTT$
               towards destination $TCPaddr_d$ as in [5];
            RTTHashTableEntry ($TCPaddr_d$) = $SRTT$;
         End If;
   End For;
   $SRTT_{UDPFlow} = 0$;
   If $UDPaddr_s$ is a key in the hash table RTTHashTable then
```

```
        $SRTT_{UDPFlow}$ = RTTHashTableEntry($UDPaddr_s$);
    End If;
    If $UDPaddr_d$ is a key in the hash table RTTHashTable then
        $SRTT_{UDPFlow}$ = $SRTT_{UDPFlow}$ + RTTHashTableEntry($UDPaddr_d$);
    End If;
    If $SRTT_{UDPFlow} = 0$ then Compute $SRTT_{UDPFlow}$ as in [2];
    End If;
End.
```

FIGURE 1. Our algorithm for improving the round trip time approximation

4. Results and Evaluation

Certain network conditions have to be met for successfully applying our algorithm. In order to detect misbehaved UDP flows, a router running our algorithm must forward TCP packets having the same destination address as the suspected flow. Although the destination address of a forwarded packet is a completely random variable, our experiments show that the proposed algorithm is suitable for an acceptable number of UDP flows.

We placed our experiments at the border router R of our Intranet. Because for any outgoing flow, the delay due to our Intranet links and routers is not a significant fraction of the flow's round-trip time, we were interested only in estimating the flow's round-trip time on the outside path of our Intranet: for a flow F as defined above, having source address $UDPaddr_s$ in our Intranet, we tried to compute the flow's round trip time on the $R - UDPaddr_d$ path. The experiments tried to determine the percent of UDP flows, for which our algorithm is suitable.

For grouping UDP packets in flows, we used a simple heuristic approach. We group in a flow UDP packets having the same source and destination addresses and the same source and destination ports. For eliminating stray UDP packets, we required that any outgoing UDP flow has a sending rate higher than ρ. We chose $\rho = 1$ *packet/second*. We also eliminated known good behaved traffic generated in our Intranet on 111, 53, 137-139 UDP ports (portmap, nameserver netbios services).

We ran our experiment ten times, in different weekdays, at different hours with our network running at different loads. For any detected outgoing UDP flow F, we looked for TCP packets and for theirs acknowledgements, going to and coming from $UDPaddr_d$ of flow F. For an average of about 6% of UDP flows we were able to detect such TCP packets.

We use for demonstrating the strength of our algorithm a simple simulation network and the following scenario. A non TCP-friendly UDP flow F, from $UDPaddr_s$ to $UDPaddr_d$, does not obey congestion control policies despite the packet drops incentives imposed by both routers R_1 and R_2. Using the approach from [2], the round trip time of flow F is estimated to twice the one-way delay of the $R_1 - R_2$ path. On a 10 Mbps path, using 1500 bytes packets, this value is around 55 milliseconds.

For successfully applying our algorithm, a competing, good-behaved TCP connection must use the bottleneck $R_1 - UDPaddr_d$ path. In these circumstances, router R_1 will successfully estimate the round trip time of this connection to 110 milliseconds, twice as the one estimated above. This value, used in a TCP-friendliness test for flow F inside the R_1 router, will adjust the sending rate of flow F to half of the sending rate allowed in the above conditions. If not properly adjusted inside the R_1 router, the sending rate of flow F will be eventually halved later, inside the R_2 router. It is better, for saving bandwidth, to drop packets of a flow earlier in the network, closer to it's source.

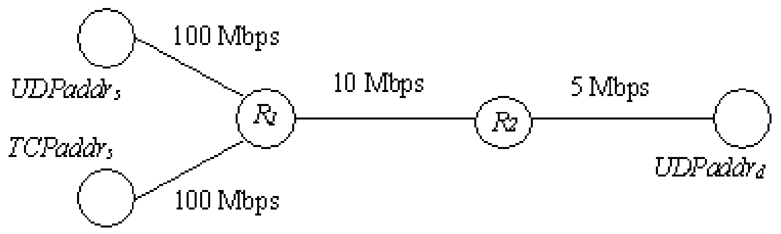

Figure 2. Simulation network

We consider that these results justify further research in this area, aiming the improvement of the percent of UDP flows that might benefit from our algorithm, or the improvement of the estimated round-trip time of an UDP flow.

5. Conclusion and Future Work

We suggest in this paper an algorithm that improves the estimated round-trip time for a reasonable number of UDP flows. As a future work we suggest running our experiments in a core router of a transit AS. Many routing algorithms use delay as one of the metrics used for computing a path costs. We also suggest, for further investigations, the use of this additional delays information retrieved from the network level for improving the estimated round-trip time of a non-TCP flow.

References

[1] Allman, M., Paxson, V. and Stevens, W., *TCP Congestion Control*, IETF RFC 2581, April 1999

[2] Floyd, S. and Fall, K., *Promoting the Use of End-to-End Congestion Control in the Internet*, IEEE/ACM Transactions on Networking, no. 4, **7**(1999), 458-472

[3] Floyd, S. and Jacobson, V., *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Trans. Net., Aug. 1993, **1**, no. 4, 397-413

[4] Floyd, S., *A Report on Some Recent Developments in TCP Congestion Control*, IEEE Communication Magazine, April 2001

[5] Jacobson, V., Braden, R. and Borman, D., *TCP Extensions for High Performance*, IETF RFC 1323, May 1992

[6] Ramakrishnan, K., Floyd, S. and Black, S., *The Addition of Explicit Congestion Notification (ECN) to IP*, RFC 3168, September 2001

[7] Ryu, S., Rump, C. and Qiao, C., *Advances in Internet Congestion Control*, IEEE Communications Surveys and Tutorials, Third Quarter 2003, **5**, No. 1, 28-39

"Babeş-Bolyai" University of Cluj-Napoca
Department of Computer Science
Mihail Kogalniceanu, No 1, Cluj-Napoca, Romania
E-mail address: {bufny, forest, claudiu, florin}@cs.ubbcluj.ro