Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. III (2008), Suppl. issue: Proceedings of ICCCC 2008, pp. 201-205

A New Method for Macroflows Delimitation from a Receiver's Perspective

Darius Bufnea

Abstract: This paper presents a new approach for shared bottlenecks detection from a receiver's perspective. This approach uses flow clustering at the receiver, based on passive observations of inter-packet arrival time intervals. We also suggest a new cost function useful in the flows clusterization process into macroflows. The proposed method can be used in the discovery of path patterns or for extending the macroflow granularity in an improved Congestion Manager.
Keywords: congestion control, bottleneck, Congestion Manager, macroflow.

1 Introduction

The latest years have brought changes in the dominant traffic type carried by the Internet infrastructure. In the mid-80s the dominant traffic was generated mainly by specialized "well-behaved" users located in universities or in research centers. Later, in the 90s, the source of the Internet traffic migrated towards business and residential users. Although, the profile of the Internet user has not changed in the last decade, the traffic profile and the amount of traffic increased dramatically. This was mainly caused by new Internet applications such as: multimedia streaming applications (video on demand over Internet, radio broadcasts, voice over IP) and peer-to-peer applications, used mainly for exchanging huge amount of data (e.g. multimedia files of considerable sizes). The traffic generated by the latest Internet applications, used mainly by unspecialized users, is carried sometimes over non congestion-aware protocols such as UDP. Consequently, the network and the transport layers must make continuous efforts to keep fairness among all Internet users, keeping their "best-effort" traffic in normal throughput patterns, while performing smooth congestion avoidance.

Congestion avoidance and control in Internet is done either at network level inside transport routers [1] or at protocol level inside a peer's TCP/IP stack [2]. It is desirable for each transport protocol to implement a congestion avoidance algorithm, or if such an algorithm is not available in the transport layer (for example the UDP transport protocol lacks a congestion avoidance algorithm), it must be implemented in the higher application layer. The congestion control at the network level is required to interfere when a peer in the communication process is not congestion-aware and the amount of data it injects into the network exceeds the amount of data carried by the network for concurrent streams.

2 Previous Work

The current congestion control mechanism, as specified and implemented by the TCP/IP stack [3] performs per-flow congestion checking. For each active TCP/IP connection, the network stack computes individually and maintains separately a series of state variables such as: congestion windows size, round trip time or retransmission time-out. The maintenance of these variables is done on a per-flow basis - there is no coordinated management of streams which compete with each other for scarce bandwidth, rather then sharing the state variables' values whose computation is often redundant. There is a proposed mechanism that implies collaboration between streams that share the same congestion parameters and often, in this situation, the same state variables values in a so called macroflow. Such a collaboration between streams would be managed by a Congestion Manger [4]. However, the problem remains in identifying the flows that must face the same congestion situation. One set of such flows are those that share the same network bottleneck.

There are some proposed techniques that detect shared bottleneck links by using additional injected traffic in the network ([5], [6]). Such an approach has the disadvantage that increases network load and depends on features that might not be available everywhere. Other techniques presented in [7] and [8] detect common bottlenecks by inspecting the time evolution of state variables and clustering flows based on this evolution. However, these techniques can be used only from the sender's point of view. In [9] the authors suggest an information based theory solution that can be used from the receiver's point of view.

In this paper, we suggest a mechanism similar to the one presented in [9] for identifying from the receiver's perspective the set of flows that share the same bottlenecks in their path towards the same destination but use a different technique for flow clustering and a different distance (similarity) measure. Such a set of flows is suitable for extending the granularity of constructing a macroflow in an improved Congestion Manager [4].

3 Mechanism and Data Model

The approach presented in this paper is suitable for detecting shared bottlenecks from a data receiver's point of view or from any transit router's point of view. The detection of shared bottlenecks is done passively by simple observation of flows' inter packet arrival time. Data packets generated by two or more sources that transmit data to a receiver over the same bottleneck, will arrive to destination at equidistant moments in time. This is because the bottleneck router has its queue full and it injects periodically, at constant length time intervals, packets in the congested output line. But these packets that transit a common bottleneck will mix up with data generated by other hosts. So, the main problem is to identify those sources that generated packets which arrive to receiver at equidistant moments in time.

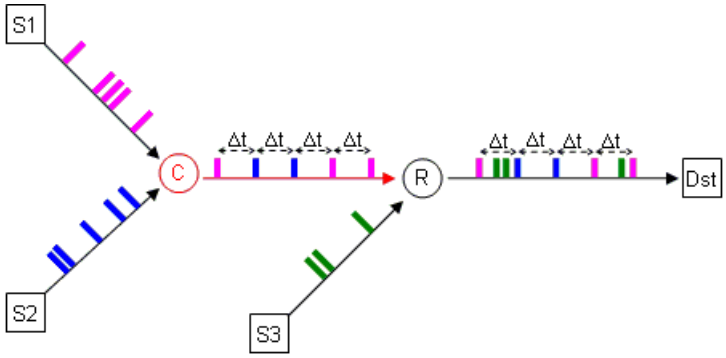

Figure 1: Equidistant inter-packet arrival time intervals

For a better understanding of the mechanism above we analyze it for the network depicted in figure 1. The S_1 and S_2 sources send packets to the *Dst* receiver over the same congested link *C*-*R*. The congested router *C* queues incoming packets and sends them over the congested link at a constant rate, each packet is output in a Δt time interval. However, this regulated traffic is mixed on the *R*-*Dst* link with the traffic generated by the S_3 source. So, at the *Dst* receiver, we have to identify those packets that reach the destination at equidistant moments in time, i.e. packets generated by the $\{S_1, S_2\}$ source set.

The data model used in this paper was presented in [9]. However, we use a modified version of the data model, adapted to achieve our goals. Let $D = \{d_1, d_2, \cdots, d_n\}$ be the set of incoming data packets at the *Dst* receiver. This set is ordered by the incoming timestamp of each packet $t_{d_1} < t_{d_2} < \cdots < t_{d_n}$ where t_{d_i} is the timestamp when *Dst* receives the d_i packet. The D packet set is generated by the source set $S = \{S_1, S_2, \cdots, S_m\}$, usually $m << n$. This assumption is obvious and respects the real traffic patterns. We denote by $Source(d_i) \in S$ the source host of packet d_i and by $\Delta t_{de} = |t_d - t_e|$, $d, e \in D$, the time interval spent between the arrival time of packet d and the arrival time of packet e. Let also F_i be the flow generated by source S_i, $T_i = \{t_{i_1}, t_{i_2}, \cdots\}$ the arrival times of flow F_i's packets and $\Delta_i = \{\delta_{i_1}, \delta_{i_2}, \cdots\}$ the inter-packet arrival time intervals of flow F_i, $\delta_{i_1} = t_{i_2} - t_{i_1}, \delta_{i_2} = t_{i_3} - t_{i_2} \cdots$. Each of the values above can be easily identified by a TCP/IP receiver by direct observation of incoming packets.

In order to simplify our model we made the assumption that all incoming packets at the receiver have the same size. For packets of different sizes, the output intervals can be normalized by the capacity of the congested link. For instance, if two different packets d and e have different sizes, B_d and B_e, those two packets will be output over the congested link at $B_d/C \approx B_e/C$ time intervals, where C is the capacity of the output link.

Identifying those packets that share the same bottleneck means determining at least a subset $D' \subset D$ of packets, $D' = \{d_{i_1}, d_{i_2}, \ldots, d_{i_k}\}$, $k < n$, ordered by arrival timestamps ($t_{d_1} < t_{d_2} < \ldots < t_{d_n}$), so that $\Delta t_{d_{i_1} d_{i_2}} \approx \Delta t_{d_{i_2} d_{i_3}} \approx \ldots \approx \Delta t_{d_{i_{k-1}} d_{i_k}}$. Also, for each such set $D' \subset D$, we can also identify $S' \subset S$ subset of sources , $S' = \bigcup_{d_{i_k} \in D'} \{Source(d_{i_k})\}$. All the flows generated by the sources in S', share the same bottleneck and will be considered part of the same macroflow.

4 Algorithm

Our main goal is to group in a cluster those flows that share the same bottleneck to the destination. Without considering the mixed data traffic, the bottleneck is reflected in a constant inter-packet arrival time intervals at the receiver. Each flow in the clustering process will be represented by its inter-packet arrival time interval vector (Δ_i), while the correct cluster will be represented by a merged vector of inter-packet arrival time intervals. We will use for clustering a hierarchical algorithm that uses as the cost function the standard deviation of a vector. For a cluster C, with a merged data vector Δ having K components, its cost is represented by:

$$\sigma_C = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(\Delta_i - \bar{\Delta})^2} \tag{1}$$

In figure 2 we present the image of a correct determined cluster vs. an incorrect one.

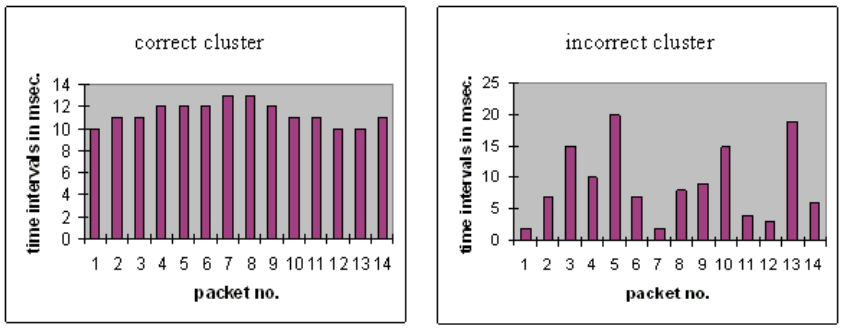

Figure 2: A correct identified macroflow vs. an incorrect one

We are using an iterative clustering algorithm, from the K-means family. The algorithm tries to group packets in clusters using the approach above, in order to reduce clusters final costs.

```
Subalgorithm ClusterIdentification is:
Input:
   m - the total number of flows that reach us;
   F_i, i = 1..m - flow i;
   T_i = {t_i1, t_i2, ...,} - the arrival times of flow F_i's packets;
Output:
   N, the number of clusters inferred in by the algorithm;
   M = {C_1, ..., C_N}, the set of inferred clusters.
Begin   // the initial clusters' configuration
   N := m; M := φ;
   For i := 1 to N do
      C_i := {F_i};
      M := M ∪ {C_i};
   endfor;
   For i := 1 to m do
      Do
         modified := false;
         C := cluster that contains flow F_i;
         min := cost_function(C);
         For each Dst cluster, Dst ∈ M, Dst ≠ C do
            If cost_function(Dst ∪ {Fi}) < min then
               Move F_i from C to Dst;
               min := cost_function(Dst);
               If C is an empty cluster then
                  M := M − C; N := N − 1;
               endif;
```

```
            modified := true;
            break;
          endif;
        endfor;
      While modified = true;
   endfor;
end; // ClusterIdentification

Function cost_function(Cluster C):real is
   Initialize an empty vector T;
   For each flow F_i ∈ C do
      Append all values from vector T_i to vector T;
   endfor;
   Ascending sort vector T by its numerical values;
   Compute merged vector inter-packet arrival intervals Δ;
   Return standard deviation σ_C of Δ;
end; // cost_function
```

5 Algorithm Evaluation

The mechanism presented in the previous section has been tested on a simulated network having the topology shown in figure 3. The infrastructure of the simulated network is a real one, but the network's links operating at 100 Mbps had been shaped using HTB [10] in order to simulate lower link capacity. Each data source opens a TCP/IP connection to the destination and sends a continuous data flow over it. The data receiver records the incoming packets together with their arrival timestamps, for each incoming flow. The arrival times were recorder over a time interval of 160 milliseconds and are presented in figure 4.

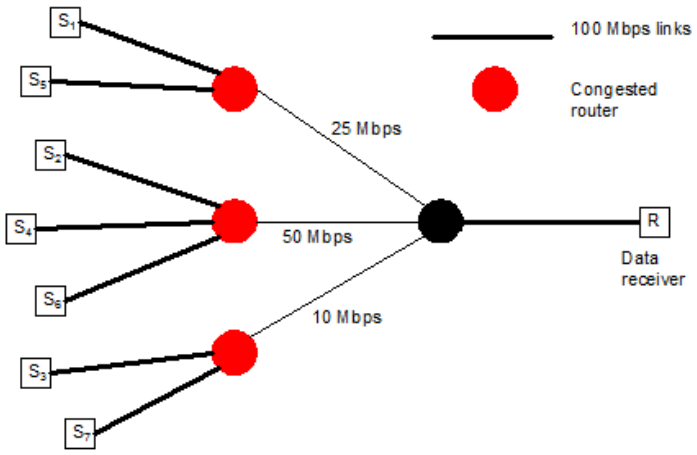

Figure 3: Simulated network

Flow	Arival times of flow's packets
F_1	7 31 55 63 85 93
F_2	4 12 22 24 38 43
F_3	10 44 56 106 117 148 159
F_4	19 26 40
F_5	15 22 39 46 70 77
F_6	7 9 15 30 33 36
F_7	21 32 68 81 94 128 138

Figure 4: Arrival times of flows' packets for the simulated network

Our method successfully detected three clusters of sources, correctly assigning the seven flows in three distinct macroflows, one macroflow for each congested link that the network contains. Figure 5 presents the identified macroflows, the flows that are part of each macroflow (cluster) and the equidistant packet arrival times for each macroflow.

Macroflow	Flows assigned to macroflow	Inter packet arrival times
C_1	F_1, F_5	7 8 7 9 8 7 9 8 7 7 8 8
C_2	F_2, F_4, F_6	4 3 2 3 3 3 4 3 2 2 4 3 3 2 2 3
C_3	F_3, F_7	10 11 11 12 12 12 13 13 12 11 11 10 10 11

Figure 5: Macroflows identified by our method for the simulated network

6 Conclusions and future work

In this paper we present a mechanism for better macroflow identification from the receiver point of view. This technique can be used inside an improved Congestion Manager to extend macroflows granularity.

The proposed mechanism successfully identified the correct macroflows when used on a simulated network. Future analysis must be performed in order to determine the performance of our method when applied to real bulk data collected from real traffic patterns. Methods for fast computation and implementation of the algortihm will be identified considering that the implementation is part of the very time-sensitive TCP/IP stack.

References

[1] S. Floyd, V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, 1(4), pp. 379-413, 1993.

[2] W. Stevens, *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery*, IETF RFC 2001, January 1997.

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

[4] H. Balakrishnan, S. Seshan, *The Congestion Manager*, IETF RFC 3124, June 2001.

[5] V. Paxson, *Measurements and Analysis of End-to-end Internet Dynamics*, Thesis Dissertation, 1997.

[6] A. Downey, *Using Pathchar to Estimate Link Characteristics*, Computer Communication Review, a publication of ACM SIGCOMM, volume 29, number 4, October 1999.

[7] D. V. Bufnea, A. Câmpan, A. S. Dărăbant, *Fine-Grained Macroflow Granularity in Congestion Control Management*, in Studia Universitatis, Vol. L(1), pp. 79-88, 2005.

[8] A. Câmpan, D. V. Bufnea, *Delimitation of Macroflows in Congestion Control Management Using Data Mining Techniques*, 4th ROEDUNET International Conference, Education/Training and Information/Communication Technologies - ROEDUNET '05, Romania, pp. 225-234, 2005.

[9] D. Katabi, I. Bazzi, X. Yang, *An Information Theoretic Approach for Shared Bottleneck Inference Based on End-to-end Measurements*, Laboratory for Computer Science, MIT, 2001.

[10] M. Devera, *Hierarchical Token Bucket Theory*, http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm, May 2002.

Darius Bufnea
"Babeş-Bolyai" University of Cluj-Napoca
Department of Computer Science
E-mail: bufny@cs.ubbcluj.ro