STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **L**, Number 1, 2005

FINE-GRAINED MACROFLOW GRANULARITY IN CONGESTION CONTROL MANAGEMENT

DARIUS VASILE BUFNEA, ALINA CAMPAN, AND ADRIAN SERGIU DARABANT

Abstract. A recent approach in Internet congestion control suggests collaboration between sets of streams that should share network resources and learn from each other about the state of the network. Currently such a set of collaborating streams - a macroflow - is organized on host pair basis. We propose in this paper a new method for grouping streams into macroflows when they behave similarly. A flow behavior is described by a set of state variables, such as the round trip time, retransmission time out or congestion window size. This extended macroflow granularity can be used in an improved Congestion Manager.

1. Introduction

Congestion control aims to control and adapt the transmission rate of the Internet streams in order to reduce the amount of dropped packets in case of overloaded communication lines and routers. Practical congestion control approaches work either at *protocol level* or at *router level*. A *transport protocol* should normally implement a congestion control algorithm. The TCP protocol, which transports over 90% of Internet data, treats this aspect. But there are other protocols, which remain congestion unaware. *Routers* have their congestion control policies and algorithms for handling congestion situations that are usually induced by misbehaved congestion unaware flows. The two mentioned approaches do not exclude each other; rather they are completing each other.

In order to properly offer reliable data transmission and congestion control, a TCP connection uses some state variables such as: the round trip time (*rtt*), the

---

Received by the editors: 15/03/2005.

2000 *Mathematics Subject Classification.* 62H30, 90B20, 68U35.

1998 *CR Categories and Descriptors.* 62H30 [**Statistics**]: Multivariate analysis – *Classification and discrimination; cluster analysis*; 90B20 [**Operations research, mathematical programming**]: Operations research and management science – *Traffic problems*; 68U35 [**Computer science**]: Computing methodologies and applications – *Information systems (hypertext navigation, interfaces, decision support, etc.)*;

retransmission time out (*rto*) [4], the congestion window (*cwnd*) and the slow start threshold (*sstresh*) [1]. Usually, each TCP connection maintains independently its own state variables and performs its own calculation for determining these variables' values.

But even when each stream, independently, incorporates congestion aware algorithms, a set of concurrent streams will still compete with each other for network resources, rather than share them effectively [2]. Recent approaches introduce the idea of Internet streams *collaborating* for an improved congestion control mechanism. Rigorous delimited (defined) set of streams should share network resources and learn from each other about the state of the network. Currently, such a set of collaborating streams, referred as a *macroflow*, is organized on host pair basis; i.e. a macroflow comprises connections sharing the same (source IP, destination IP) pair. We propose in this paper a new method for grouping streams into macroflows according to their similar behavior. This method provides an accurate, less naive approach for delimiting macroflows inside the overall set of connections maintained by a host. As a consequence, more connections will be detected as being part in one macroflow and will share their network knowledge. This approach is meant to be part of an improved congestion Control Manager.

1.1. **Related Work.**

Floyd suggested in [6] that the *rtt* and *rto* values should be the same for all connections that share the same (source IP, destination IP) pair in the same moment in time. For this reason, she claimed that the network level should be maintaining the values of these state variables, and not the transport level. However, Floyd did not further explore this approach.

[5] joins the idea of sharing state variables between flows, on host pair basis. In addition, she gives practical suggestions and solutions for accomplishing this, in certain concrete situations.

[3] describes the LINUX caching mechanism of state variables values. One set of information is maintained for each destination IP. The cached values serve for state variables initialization of new connections targeting the same destination IP. Thus, the LINUX caching mechanism also functions on host pair basis.

A state of the art approach [2] in congestion control suggests a practical way for the collaboration between transport protocols and applications. This collaboration should take place into an integrated *Congestion Manager* (CM) framework. All protocols and applications involved in such a framework should provide their network knowledge (*rtt*, packet losses) to the CM. The CM should aggregate all these information on host pair basis (macroflow basis), "learn" from them and inform the protocols and applications, in a synchronous or asynchronous manner, about when and how much data they can safely "put on the wire". Practically, the collaboration will take place, mediated by the CM, between connections inside a macroflow; no collaboration will happen across macroflows. So, more adequate the macroflows are established, more efficient the CM's control will be.

1.2. **Contributions.** We propose in this paper a new method for grouping streams into macroflows when they behave similarly. A flow behavior is defined by a set of state variables, such as the round trip time, retransmission time out or congestion window size. The advantage is that we can cluster together streams not only on host pair basis, but also on LAN pair basis; even more generally, streams sharing a particular network bottleneck will be identified by our method. This extended macroflow granularity can be used in an improved Congestion Manager.

2. Data Model

2.1. ***Rtt* Vectors.** We consider the case of an upload server that treats a high number of simultaneous incoming connections. The aim is to establish (infer) inside this set of connections some groups of connections with similar behavior. A Congestion Manager running on that server will treat such a group as a macroflow.

We denote by S the *server machine* itself or its *network identification IP address.*

Each incoming connection is identified by the server S by a pair $(C_{IP} : C_{port})$, where C_{IP} is the client's IP address and C_{port} is the client's port identification.

During the life time of each $(C_{IP} : C_{port})$ connection, the server S will periodically measure and retain the values of some state variables, such as the round trip time, retransmission time out or congestion window size. We based our experiments on measurements of the round trip time (*rtt*) state variable. Practical ways for the achievement of measurements are described in [8, 9].

Therefore, from the point of view of the upload server S, the incoming connection $f = (C_{IP} : C_{port})$ during the time interval (t_b, t_e) is described by the *rtt vector* $V = (r_1, r_2, \ldots, r_k)$ where:

- $(t_b, t_e) \subseteq (C_{IP} : C_{port})$ connection life time;
- Δt is the interval between two consecutive measurements;
- $k = (t_e - t_b)/\Delta t$;
- r_i is the *rtt* value measured at the $t_b + \Delta t * (i - 1)$ time moment.

We say that the *rtt vector* associated to a connection describes the connection's behavior. The choice of the *rtt* state variable for describing a connection behavior is justified as follows. For two connections f_1 and f_2 coming from the same client or LAN the *rtt* values measured at the same moment in time are quasi-identical. Therefore, their associated *rtt vectors* during the same time interval are also quasi-identical. This means that f_1 and f_2 manifest a similar behavior, which justifies their placement in the same macroflow. As we said, we want to extend the macroflow granularity outside the host-pair scope, on the basis of similar behavior; but we also want to keep such connections (as f_1 and f_2 are) as much as possible together in macroflows. So, according to our modeling such connections must have similar behavior. The rtt vectors model ensures that fact.

2.2. **Similarity Measure.** For grouping similarly behaving flows we will use, as described in the next paragraph, an artificial intelligence clustering algorithm. Such an algorithm needs a similarity measure and a distance function for comparing and differentiating two analyzed flows. We propose next such measures and justify our choice.

We associated to a connection an *rtt vector* describing its behavior. The *rtt vector* reflects the *rtt* temporal evolution of that flow. Two connections will be considered more similar as they are more linearly correlated (e.g. the *rtt* values for the two connections increase and decrease at the same moments in time). A statistical measure for the linear correlation of two vectors is the *Pearson coefficient.*

Given two connections, $f_1 = (C^1_{IP} : C^1_{port})$ and $f_2 = (C^2_{IP} : C^2_{port})$ measured during the time interval (t_b, t_e) and their associated *rtt vectors* $V_1 = (r_{11}, r_{12}, \ldots, r_{1k})$ and $V_2 = (r_{21}, r_{22}, \ldots, r_{2k})$, the *Pearson correlation coefficient* of f_1 and f_2 is defined as:

$$P(V_1,V_2) = \frac{\sum_{i=1}^{k} r_{1i}\cdot r_{2i} - \frac{\sum_{i=1}^{k} r_{1i}\cdot\sum_{i=1}^{k} r_{2i}}{k}}{\sqrt{\left(\sum_{i=1}^{k} r_{1i}^2 - \frac{(\sum_{i=1}^{k} r_{1i})^2}{k}\right)\cdot\left(\sum_{i=1}^{k} r_{2i}^2 - \frac{(\sum_{i=1}^{k} r_{2i})^2}{k}\right)}}$$

$$= \frac{\sum_{i=1}^{k}(r_{1i}-\overline{r_1})\cdot(r_{2i}-\overline{r_2})}{\sqrt{\left(\sum_{i=1}^{k}(r_{1i}-\overline{r_1})^2\right)\left(\sum_{i=1}^{k}(r_{2i}-\overline{r_2})^2\right)}} \tag{1}$$

where $\overline{r_1}$ and $\overline{r_2}$ are the mean values of V_1 and V_2.

$P(V_1, V_2)$ takes values in [-1,1] interval, a value of 1 meaning that the compared vectors are linearly correlated, and a value of -1 meaning that they are inversely linearly correlated. We chose to transport the Pearson coefficient values in [0,1] interval. However, this will not affect the semantics of the transformed coefficient.

Its maximal value will still indicate o positive correlation between parameters, and the minimal value a negative correlation. Therefore, the similarity measure we use for comparing connections will be:

$$\overline{P}(V_1, V_2) = \frac{P(V_1, V_2) + 1}{2} \tag{2}$$

For differentiating connections we use a distance function defined by:

$$d_P(V_1, V_2) = 1 - \overline{P}(V_1, V_2) \tag{3}$$

d_P take values in [0,1]; two identical flows will be at 0 distance, two negatively correlated flows will be separated by a distance of 1.

The distance function defined on the basis of the correlation coefficient as above does not satisfy the triangle inequality; it is, therefore, what is called a *semi-metric*.

We have to notice a shortcoming of the correlation coefficient in describing vectors with a constant evolution (e.g. vectors with all the components equal). If one of the vectorial arguments of P is constant, the correlation coefficient is undefined. We choused to consider it -1, as nothing can be said about the correlation between such arguments and we want them not to disturb the classification of the other connections with well defined behavior.

3. The *MacroflowIdentification* Algorithm

Let $F = f_1, f_2, \ldots, f_n$ be the set of all incoming concurrent connections served by S. For the (t_b, t_e) time interval, the measured *rtt vectors* are $V = V_1, V_2, \ldots, V_n$, where V_i is the *rtt vector* associated to f_i, $f_i = (C^i_{IP} : C^i_{port}), V_i = (r_{i1}, r_{i2}, \ldots, r_{ik})$.

We use an *agglomerative hierarchical clustering algorithm* for grouping in macroflows the concurrent connections described by similar *rtt vectors*. This bottom-up strategy starts by placing each connection in its own cluster (macroflow) and then merges these atomic clusters into larger and larger clusters (macroflows) until a *termination condition* is satisfied.

At each iteration, the closest two clusters (macroflows) are identified. The distance between two clusters M_i and M_j is considered to be, as defined in (4), the maximum distance of any pair of objects in the cartesian product $M_i \times M_j$. If the distance between these two closest clusters does not exceed a given threshold *thr_max_dist*, we merge them and the algorithm continues by a new iteration. Otherwise, the algorithm stops. So, the *termination condition* holds when there are no more clusters closer than a given threshold.

This decision regarding the termination condition is justified. We want that the resulting macroflows do not contain *any* "wrong" placed connections, so that the subsequent decisions based on our macroflow delimitation not to be erroneous. A macroflow is "correct" if any pair of its objects are similar enough.

The threshold *thr_max_dist* was chosen above 0.95, to ensure correct macroflow construction. By merging two clusters that are close enough with respect to the threshold *thr_max_dist* ensures that, inside the obtained merged cluster (macroflow), any two connections are not more distant than *thr_max_dist*. So, it is safe to place them into the same macroflow.

```
Algorithm MacroflowIdentification is
Input:  n, the number of concurrent connection at server S;
        $F = \{f_1, f_2, \ldots, f_n\}$ the set of concurrent connection at S;
        $V = \{V_1, V_2, \ldots, V_n\}, V_i = (r_{i1}, r_{i2}, \ldots, r_{ik})$, $i = 1..n$, the rtt vectors
        associated to the connections;
        thr_max_dist, the maximal distance threshold for two connections
        to be admitted in the same macroflow.
Output:m, the number of macroflows inferred in the concurrent connections
        set;
        $M = \{M_1, \ldots, M_m\}$, the inferred macroflows, where
        $M_i \neq \varnothing, i = 1..m$, $\cup_{i=1}^{m} M_i = F$, $M_i \cap M_j = \varnothing, i, j = 1..m, i \neq j$.
Begin
    m := n;
    $M$ := $\varnothing$;
    For i:= 1 to m do
       $M_i := \{f_i\}$;
       $M := M \cup \{M_i\}$;
    End For;
    While (m>1) and (Continue($M$,thr_max_dist,$M_{merge1}, M_{merge2}$)=true) do
       $M_{new} := M_{merge1} \cup M_{merge2}$;
       $M := M - \{M_{merge1}, M_{merge2}\} \cup \{M_{new}\}$;
       m := m-1;
    End While;
End.


Function Continue($M$ the set of current macroflows, thr_max_dist,
                  out $M_{merge1}$, out $M_{merge2}$):boolean is
Begin
```

```
    min_dist := $\infty$;
    For each $M_i \in M$
       For each $M_j \in M, M_j \neq M_i$
(4)       $dist(M_i, M_j) = max\{d_P(v_r, v_t) | f_r \in M_i, f_t \in M_j\}$;
          If $dist(M_i, M_j) < min\_dist$
             $min\_dist := dist(M_i, M_j)$;
             $M_{merge1} := M_i; M_{merge2} := M_j$;
          End If;
       End For;
    End For;
    If $min\_dist < thr\_max\_dist$ Return True;
    Else Return False;
    End If;
End Function.
```

Function Continue determines the closest two clusters from the clusters set M. It will return true if these clusters are closer than thr_max_dist and false otherwise.

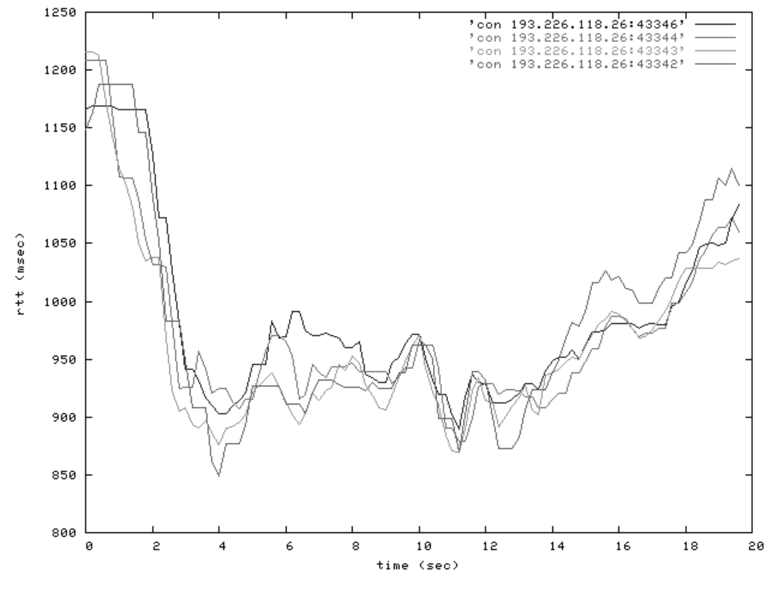

FIGURE 1. A macroflow composed of connections originating from the same client IP

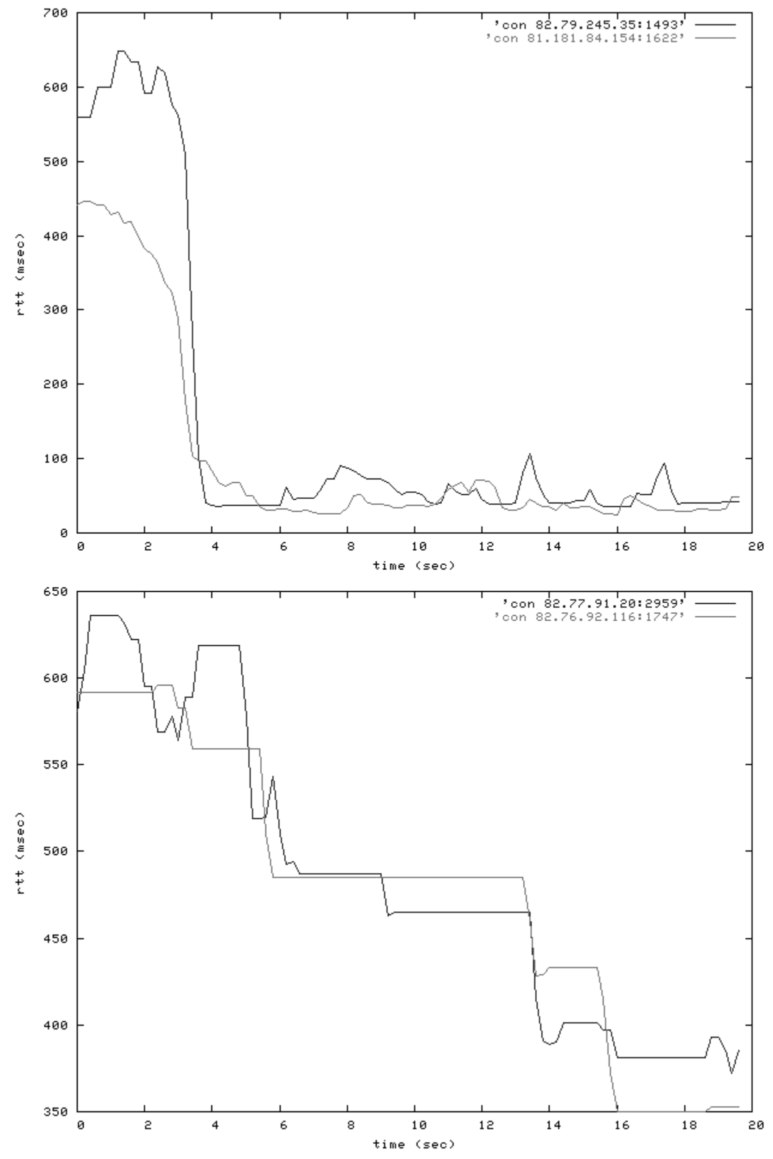

FIGURE 2. Macroflows composed of connections originating from different client IPs

4. Results and Evaluation

To test the efficiency of the proposed algorithm, we used it on an http upload server, with a high number of incoming connections. We measured the *rtt* state variable for the incoming connections at S during a larger time interval and we considered for clustering samples of 20 seconds. We take the case of one such sample, composed of 89 connections, originated from 50 different remote hosts. Inside this connection set our algorithm detected 38 macroflows.

The connections originating from the same client were, most of them, clustered together in one or few clusters. Figure 1 represents a macroflow in which were grouped together 4 connections with the same client IP.

But the algoritm also detected, as we intended, macroflows over connections coming from different client IPs. Two such macroflows are illustrated in Figure 2. It can be clearly seen the similar *rtt* evolution during time for the connections of each macroflow. This fact might happen in different situations: client IPs hosted in the same remote LAN or client IPs sharing the same bottleneck toward server S.

The connections with quasi-constant (almost all components of the associated *rtt vectors* are equal) were all grouped together in one cluster - however, taking into account the behavior of the Pearson correlation coefficient in the presence of a constant vector argument they are not to be considered as similar and forming a macroflow.

5. Conclusions and Future Work

We suggested in this paper a data model and an algorithm for extending the macroflow granularity outside of the host-pair approach. Our method will prove its advantages in a Congestion Manager framework.

As a future work we plan to explore the use of different similarity measures and other state variables to compare the timely evolution of the connections being analyzed. We also want to extend the clustering method to an incremental variant having the ability to deal with new connections entering or leaving the system at any given moment.

References

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

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

[3] Sarolahti, P., Kuznetsov, A. - Congestion Control in Linux TCP, In Proc. of the FREENIX Track: 2002 USENIX Annual Technical Conference, pp 49-62, 2003.

[4] Paxon, V., Allman, M. - Computing TCP's Retransmission Timer, IEFC RFC 2988, November 2000.

[5] Touch, J. - TCP Control Block Interdependence, IETF RFC 2140, April 1997.

[6] Floyd, S. - A report on Some Recent Developments in TCP Congestion Control, IEEE Communication Magazine, 39(4), 2001.

[7] Han, J., Kamber, M. - Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, 2001.

[8] Bufnea, D., Sterca, A., Cobarzan, C., Boian, F. - Improving the Round Trip Time Estimation in Internet Routers, In Carpathian Journal of Mathematics, 20(2), Proc of the 4th Intl. Conf. on Applied Mathematics (ICAM4), Baia Mare, Romania, pp 149-154, 2004.

[9] Bufnea, D., Sterca, A., Cobarzan, C., Boian, F. - TCP State Variables Sharing, Proc. of the Symposium Zilele Academice Clujene, Romania, 2004.

Babes Bolyai University, Cluj Napoca,Romania
E-mail address: `bufny@cs.ubbcluj.ro`

Babes Bolyai University, Cluj Napoca, Romania
E-mail address: `alina@cs.ubbcluj.ro`

Babes Bolyai University, Cluj Napoca,Romania
E-mail address: `dadi@cs.ubbcluj.ro`