DELIMITATION OF MACROFLOWS IN CONGESTION CONTROL MANAGEMENT USING DATA MINING TECHNIQUES

Alina CÂMPAN, Darius BUFNEA
Department of Computer Science, Faculty of Mathematics and Computer Science,
"Babeş-Bolyai" University of Cluj-Napoca

Abstract: *Some of the newest approaches in Internet congestion control management suggest collaboration between sets of streams that should share network resources and learn from each other about the state of the network. A set of such collaborating streams is called a macroflow. In classical congestion control approach, a stream learns information about the network state by itself. It makes use of the acquired knowledge to adapt its transmission rates to the current network conditions. Stream collaboration, in exchange, permits dissemination of network state knowledge: the streams in a macroflow maintain common information about the network state. Every stream in that macroflow uses this shared knowledge and contributes to its maintenance. This dissemination of network knowledge conducts to a better, faster and more flexible adaptation of flow behavior in presence of network congestion. The remaining problem is how to identify the streams forming such a logical entity (a macroflow). Currently a macroflow is organized on host pair basis. We propose in this paper a new method for grouping streams into macroflows if they behave similarly. A flow behavior is described by a set of state variables, such as the congestion window size, the round trip time or retransmission time out. This extended macroflow granularity can be used in an improved Congestion Manager.*

Key words: *congestion control, Congestion Manager, macroflow, and congestion window*

1. Introduction

Congestion control aims to control and adapt the transmission rate of the Internet streams so that to reduce the amount of dropped packages 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 (such as UDP), 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 retransmission time out (*rto*) [PA00], the congestion window (*cwnd*) and the slow start threshold (*sstresh*) [AP99, JV88]. Usually, each TCP connection maintains independently its own state variables and performs its own calculation for determining these variables' values.

But even if each stream, independently, incorporate congestion aware algorithms, a set of concurrent streams will still compete with each other for network resources, rather than share them effectively [BS01]. 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, better-chosen 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

[FL01] suggested 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, [FL01] did not further explore this approach.
[T097] joins the idea of sharing state variables between flows, on host pair basis. In addition, she gives practical suggestions and solutions for accomplishing this sharing, in certain concrete situations.
[SK02] 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 [BS01] 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 (*cwnd*, *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, by the mediation of 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 if they behave similarly. A flow behavior is defined by a set of state variables, such as the congestion window size, the round trip time or retransmission time out. 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

We consider the case of an upload server that treats at the same time a high number of incoming connections. The aim is to establish 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 congestion window size, the round trip time or retransmission time out. We based our experiments in this paper on measurements of the congestion window size (*cwnd*) state variable. Practical ways for the achievement of measurements are described in [B+04].

2.1. Cwnd Vectors

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 *cwnd vector* $V = (r_1, r_2, \ldots, r_k)$ where:

- (t_b, t_e) $\subseteq$ (C_{IP}:C_{port}) connection life time;
- Δt is a fixed time interval (in experiments around 100 milliseconds);
- $k = (t_e - t_b)/\Delta t$;
- r_i = 0 if the congestion window size decreased at least once during the time interval T_i =[$t_b+\Delta t*(i-1)$, $t_b+\Delta t*i$), and r_i = 1 otherwise (e.g. the congestion window size increased or remained constant during that time interval), $1 \leq i \leq k$.

We say that the *cwnd vector* associated to a connection describes the connection's behavior.

The choice of the *cwnd* state variable for describing a connection behavior is justified as follows. For a connection *f*, the congestion window size represents its own estimation about the network's available transport capacity. A decrease of the congestion window size occurs when a congestion situation appears along the network path from *S* towards the destination host. If, during a larger time interval, the congestion window size decreases for two connections f_1 and f_2 in approximately the same time this means that congestion happens for both of them together, in the same moments. So it is very likely that these two connections share a bottleneck. For this reasons, it is justified to place f_1 and f_2 in the same macroflow. Consequently, they will maintain and use common information about the network state. If one of them will ever face a congestion situation (packets loss), the other one will use this information and act accordingly without having to experience this crisis by itself.

2.2. Similarity Measure

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

We associated to a connection a *cwnd vector* describing its behavior. The *cwnd vector* reflects the *cwnd* timely evolution of that flow. Two connections will be considered more similar as they meet congestion together more often. We express next the similarity of two given 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), in terms of their associated *cwnd vectors* $V_1 = (r_{11}, r_{12}, \ldots, r_{1k})$ and $V_2 = (r_{21}, r_{22}, \ldots, r_{2k})$.

Definition 1. Given a *radius step*, which is an integer number, $0 \leq step << k$, and a time interval $T_i = [t_b + \Delta t *(i-1), t_b + \Delta t *i)$, $1 \leq i \leq$ k we call f_1 and f_2:

a) *congestion neighbors* on interval T_i iif either:

- $r_{1i}=r_{2i}=0$, which means that during T_i both streams faced congestion;

or

- $r_{1i} \neq r_{2i}$ and $r_{di} = 0$, $d \in \{0,1\}$ and $\exists j$, $\max\{1, i\text{-}step\} \leq j \leq \min\{k, i+step\}$ so that $r_{1\text{-}d,\, j} = 0$.

These conditions ensure that when one of the streams f_1 and f_2 faces congestion, the other stream also faces congestion in the same interval T_i, or in one of the neighbor intervals $T_{i\text{-}step}, \ldots, T_{i+step}$; of course we take into consideration only intervals that exist: $i\text{-}step \geq 1$ and $i+step \leq k$. The similarity between two streams will increase with the number of intervals T_i where they are *congestion neighbors*. This definition takes into consideration the sequentiality in detecting congestion for two different streams at server *S*. Sending and receiving packets, measuring state variables for more connections, congestion detection by different streams are in most cases actions performed by a single kernel dealing with the TCP/IP stack and, in most cases, a single working processor. These facts, associated with the temporal discretization we have performed in our experiments, has conducted to the necessity of allowing congestion to be reflected, for two streams, not by the same vector components r_{1i} and r_{2i}, but by two close in time measurements.

b) *congestion disassociated* on interval T_i iif $r_{1i} \neq r_{2i}$ and $r_{di} = 0$, $d \in \{0,1\}$ and $not \exists j$, $\max\{1, i\text{-}step\} \leq j \leq \min\{k, i+step\}$ so that $r_{1\text{-}d,\, j} = 0$. The similarity between two streams will be decreased with the number of intervals T_i where they are *congestion disassociated*.

Definition 2. Given a *radius step*, which is an integer number, $0 \le step << k$ we define for f_1 and f_2 the following sets:

a) $CN(V_1,V_2) = \{i \mid 1 \le i \le k, f_1 \text{ and } f_2 \text{ are congestion neighbours on } T_i\}$;

b) $CD(V_1,V_2) = \{i \mid 1 \le i \le k, f_1 \text{ and } f_2 \text{ are congestion deassociated on } T_i\}$.

Definition 3. Given a *radius step*, which is an integer number, $0 \le step << k$ the ***congestion similarity coefficient*** of f_1 and f_2 is defined as:

$$CS(V_1,V_2) = \frac{|CN(V_1,V_2)| - |CD(V_1,V_2)|}{|CN(V_1,V_2)| + |CD(V_1,V_2)|}, \text{if } |CN(V_1,V_2)| + |CD(V_1,V_2)| > 0, \text{otherwise} \quad (1)$$
$$CS(V_1,V_2) = -1$$

$CS(V_1,V_2)$ describes the similarity of f_1 and f_2 with respect to congestion occurrence. It takes values in [-1,1] interval. Higher is the *CS* measure more similar are the compared vectors.

Definition 4. For differentiating connections we use a ***congestion distance function*** defined by:

$$d_C(V_1,V_2) = \frac{1 - CS(V_1,V_2)}{2} \quad (2).$$

d_C 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.

3. 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 *cwnd vectors* are $V=\{V_1, V_2, \ldots, V_n\}$, where V_i is the *cwnd 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 ([HK01]) for grouping in macroflows the concurrent connections described by similar *cwnd 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 (3), the maximum distance of any pair of objects in the cartezian 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* is that there are no more clusters closer than a given threshold.

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

The threshold *thr_max_dist* was chosen above 0.8, 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 objects (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;
$F=\{f_1, f_2, \dots, f_n\}$ the set of concurrent connection at S;
$V=\{V_1, V_2, \dots, V_n\}$, $V_i=(r_{i1}, r_{i2}, \dots, r_{ik})$, $i=1..n$, the cwnd 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;
$M = \{M_1, \dots, M_m\}$, the inferred macroflows, where

$$M_i \neq \phi,\ \ i=1..m,\ \ \bigcup_{i=1}^{m} M_i = F,\ \ M_i \cap M_j = \phi,\ \ i,j=1..m,\ \ i \neq j$$

Begin

```
m := n;
M := ∅;
For i:=1 to m do
  Mi := {fi};
  M := M ∪ {Mi};
End For;
```

```
  While (m>1) and (Continue(M,thr_max_dist,M_merge1,M_merge2)=true) do
    M_new := M_merge1 ∪ M_merge2;
    M := M − {M_merge1, M_merge2} ∪ {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
  min_dist := ∞;
  For each M_i∈M
    For each M_j∈M, M_j≠M_i
      dist(M_i, M_j) = max{d_C(v_r, v_t) | f_r ∈ M_i, f_t ∈ M_j};        (3)
      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.

4. Results and Evaluations

To test the efficiency of the proposed algorithm, we used it at an http upload server, with a high number of incoming connections. We measured the *cwnd* state variable for the incoming connections at *S* during a larger time interval and we considered for clustering samples of 40 seconds. We take the case of one such sample, which comprised a large number of connections. We present in Figure 1 two macroflows that our algorithm detected inside this connection set. There were detected, as we intended,

macroflows over connections coming from different client IPs. It can be clearly seen the similar *cwnd* 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. This extended macroflow granularity can be used in an improved Congestion Manager.

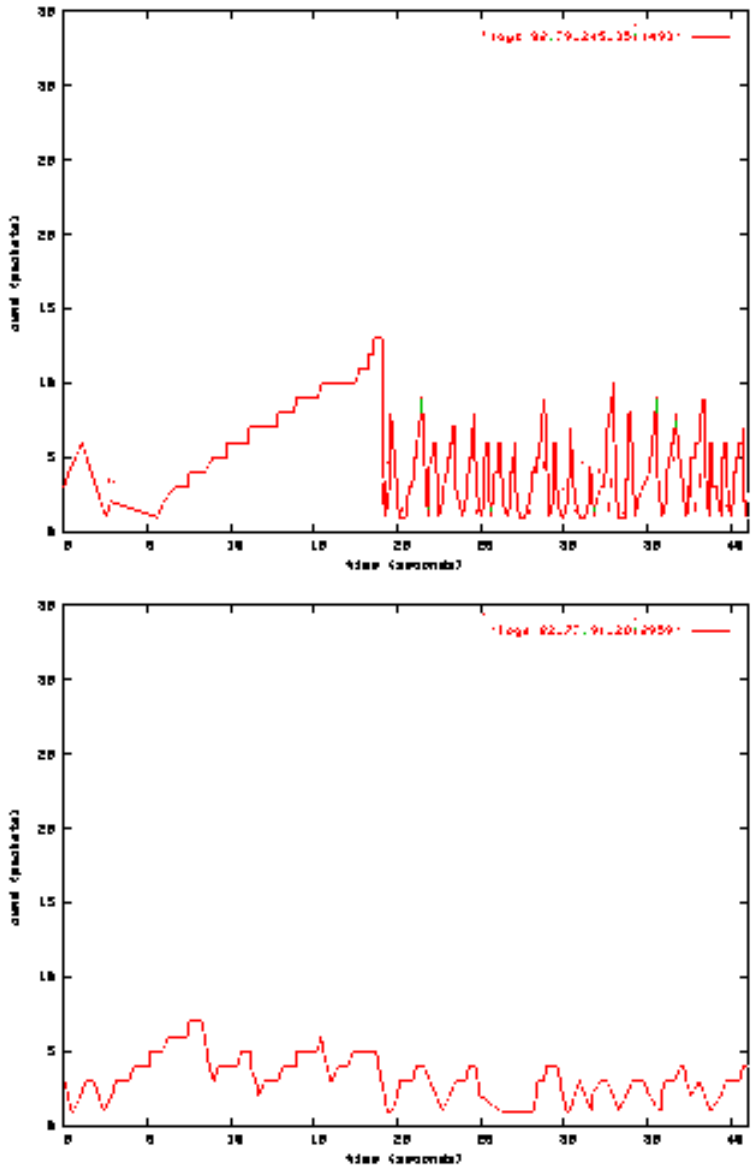

Figure 1. Macroflows composed of connections originating from different client IPs

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 adapt macroflows according to observations of streams' evolution during more than one surveillance intervals. The incremental mechanism also has to deal with new connections entering or leaving the system at any given moment.

The collaboration strategy and mechanisms for the streams inside a macroflow will also be subject to study, as they are influential factors for the stream behavior. Consequently, the macroflow delimitation mechanisms (incremental or no) will have to consider these behavioral effects induced by the collaboration mechanisms.

References

[AP99] Allman, M., Paxson, V., Stevens, W., *TCP Congestion Control*, IETF RFC 2581, April, 1999.

[BS01] Balakrishnan, H., Seshan, S., *The Congestion Manager*, IETF RFC 3124, June 2001.

[B+04] Bufnea, D., Sterca, A., Cobarzan, C., Boian, F., *TCP State Variables Sharing*, Proc. of the Symposium "Zilele Academice Clujene", pp 129-133, Cluj-Napoca, 2004.

[FL01] Floyd, S., *A reports on Some Recent Developments in TCP Congestion Control*, IEEE Communication Magazine, April 2001, vol 39, no 4.

[HK01] Han, J., Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

[JV88] Jacobson, V., *Congestion Avoidance and Control*, Proc. of SIGCOMM '88, ACM, 1988.

[PA00] Paxon, V., Allman, M., *Computing TCP's Retransmission Timer*, IEFC RFC 2988, November 2000.

[SK02] Sarolahti, P., Kuznetzov, A., *Congestion Control in Linux TCP*, Proc. of the FREENIX Track, USENIX Annual Technical Conf., pp 49-62, 2002.

[T097] Touch, J., *TCP Control Block Interdependence*, IETF RFC 2140, April 1997.