УДК 004.75

O. V. Hordiichuk[1], postgraduate

## A congestion control algorithm for video multicasting in peer-to-peer networks

[1] Taras Shevchenko National University of Kyiv, 83000, Kyiv, Glushkova st., 4d,

e-mail: oleg.gordiychuck@gmail.com

Гордійчук О. В.[1], аспірант

## Алгоритм контролю перевантажень для широкомовної передачі відео в однорангових мережах

[1] Київський національний університет імені Тараса Шевченка, 83000, м. Київ, пр-т. Глушкова 4д,
e-mail: oleg.gordiychuck@gmail.com

*Перевантаження комп'ютерних мереж виникає внаслідок відправки пакетів, кількість яких перевищує пропускну здатність напряму передачі. Через це певна кількість пакетів не доходить до одержувача або надходить із значною затримкою. Для вирішення цих проблем використовуються алгоритми контролю та уникнення перевантажень в мережах, які застосовуються у протоколах прикладного рівня.*

*В даній статті пропонується новий алгоритм контролю перевантажень, який передає дані якомога більшій кількості отримувачів і в той же час мінімізує кількість втрачених пакетів та затримку передачі, що є принциповим моментом для відеотрансляцій. На відміну від існуючих розробок даний алгоритм використовує інформацію про перевантаження одразу від усіх отримуючих вузлів, що надає змогу більш ефективно використовувати мережу за рахунок об'єктивної оцінки навантажень. Також запропонований підхід не потребує додаткової затримки перед відправкою кожного пакету, що покращує якість сервісу для кінцевих користувачів та надає змогу використовувати його в однорангових мережах деревоподібного типу для відео трансляцій.*

*Ключові слова: контроль перевантажень, уникнення перевантажень, однорангові мережі, широкомовна передача.*

*A network congestion occurs due to transmitting amount of packets that is bigger than link's bandwidth. That's why some of packets are lost or arrived with significant delay. Congestion control and avoidance algorithms are used for solving these problems and implemented as application layer protocols.*

*In this paper proposed a new congestion control algorithm that sends data to as much as possible receivers and at the same time an amount of lost packets and transmission delay are minimized that is very important for video streaming. Such approach provides more efficient bandwidth utilization by using impartial assessment of the network load unlike existing solutions and also it doesn't need additional delay before sending every packet that improves the QoE (quality of experience) for end users and this enables its usage in a tree-like peer-to-peer networks for video streaming.*

*Key Words: congestion control, congestion avoidance, peer-to-peer networks, multicasting.*

Статтю представив д.т.н., проф. Погорілий С.Д.

### Introduction

The network congestion is an old problem that appeared at the same time as the Internet. It's happened due to limited capabilities of the routing hardware that transmits data with abnormal sending rate. If transmission rate exceeds hardware limits then each new packet is pushed to a waiting queue and when it's full then packet is dropped. That's why there is a need for an algorithm that provides efficient and fast data transmission.

There are different schemes for solving this problem that implemented in network protocols. The most widespread is a congestion control in TCP\IP. It has been improving during developing of the Internet and that's why there are lot of implementations like Tahoe and Reno, Vegas, CUBIC, Westwood+ and others [1]. The main idea of this congestion control algorithm is to provide fast data transmission with fair network bandwidth usage among all applications. All implementations of TCP\IP congestion avoidance mechanism use only

the packet loss as an indicator of the network overloading. But this approach is not good for all network types. For example, LTE, 3G and ADSL routers have big waiting queues and in this case congestion detection happens much later (depending on the queue size) than a real event occurs. The algorithm that proposed in this paper also uses the packet loss event as the indicator of the network congestion, but on the other hand it reacts more smoothly.

A solution of the big queue problem was proposed in LEDBAT [2] scheme that is used in uTP. In this approach each time a sender transmits new packet it stores current machine time $t_1$ in microseconds in the special field. After packet arrives the receiver calculates difference $d = t_1 - t_2$, where $t_2$ is a current time and sends this value back to the sender. Although clocks of the sender and the receiver are not synchronized and this task is not even solvable with microseconds accuracy, the minimum value of $d$ ($d_{min}$) represents a situation when the waiting queue is empty. That's why if compare each new value $d$ with $d_{min}$ it's possible to estimate router's queue size. The LEDBAT uses this heuristic for the congestion control and tries not to exceed the queue size for more than 100ms. The solution described here also uses such approach for detecting the network congestion.

There is also a hybrid solution that is called TCP friendly rate control (TFRC) [3]. It uses a throughput equation that is based on different parameters like a round-trip time, a loss event rate, a retransmission timeout and others. This protocol is much smoother than classical TCP congestion control.

Besides protocols described above there are different other congestion avoidance solutions like DCCP [4] that can use different algorithms and Sprout [5] that uses stochastic forecasts for achieving high throughput and low delay, but only for cellular networks, where big waiting queue are used. In this paper considered the algorithm that rely on receiving speed rather on forecasts because it has more accurate information about network congestion due to receiving information from different sources.

Although there were different attempts to solve this problem, all of them are not well suitable for live streaming in peer-to-peer networks because of their delivery time constraints. This paper proposes a new congestion control algorithm that has a goal to transmit as much as possible data for the shortest period of time. It doesn't provide fairness for other streams and tries to capture all available bandwidth, however at the same time it doesn't overload router and keeps the waiting queue free.

## Congestion detection and virtual network queue

As it was mentioned before the network congestion occurs when routers traffic exceeds its bandwidth capabilities. The main indicator of network congestion is a packet loss. Each router maintains its queue using some algorithm such as random early detection [6], weighted random early detection (an improved type of previous algorithm), Blue [7] or tail-drop [8] that are called in general as active queue management. The tail-drop approach drops all packets that couldn't be pushed into the queue. Random early detection drops random packets instead of those that come last. It helps to make data transmission fairer as congestion will be detected earlier than the waiting queue becomes full.

A variety of the queue management mechanisms is a main problem for any congestion avoidance algorithm as it's impossible to detect what kind of algorithm is used in current situation. An only possible way to detect network congestion is to collect data about lost packets and transmission delay. But not all lost packets indicate the congestion as in all types of wireless networks (wi-fi, bluetooth, EDGE, 3G and LTE) this event may occur due to physical characteristics of an environment. That's why classical TCP\IP implementations as well as other algorithms have poor performance in networks where errors are possible. The same problem concerns the network delay estimation. Not all routers have so big waiting queue that congestion algorithm could detect it and at the same time doesn't garble it with a network noise. Again the wireless networks may not have stable environment and measurement of delay may be inaccurate. For example, node with EDGE modem may have difference between upper and lower bound of the round trip time up to several seconds. In consequence to these facts there is no detection scheme that works excellent in all possible cases of the network congestion.

A possible solution is to use another indicator of the congestion that doesn't affect the packet loss problem. It's known that congestion occurs when sending rate is bigger than receiving rate and usage of this fact is enough for a correct guess. But in fact we can't estimate sending and receiving rate and make correct assumption because this information holds on opposite side. Information about receiving rate should be sent back to the sender. Due to the fact that round trip time could reach hundreds of

milliseconds this information couldn't be used in such way.

The congestion algorithm may use a virtual queue instead. The sender puts every packet into its virtual queue and it holds there until the acknowledgement packet (ACK) is received. In case when the acknowledgement hasn't received in time (the packet's age reached a retransmission timeout, RT), the sender erases the packet from the queue and resends it again. This process is described in figure 1 and 2. This is very similar to a sending window in TCP\IP congestion control mechanism, but unlike "window" approach the virtual queue doesn't have limits in size and sender doesn't increase or decrease it. Instead it just simulates a real situation at the queue of router that is a bottleneck of the transmission.

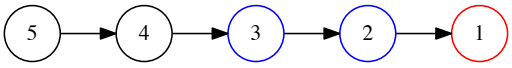


Figure 1. The virtual queue, where blue colored packets represent situation when sender has received an acknowledgement for them and red-colored is a lost packet.

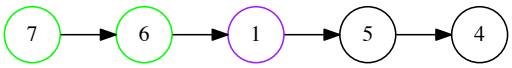


Figure 2. The virtual queue after previous state. Here the lost packet (marked as purple) is erased from the head and pushed to the tail and green-colored packets are new coming.

### Local congestion control

While the virtual queue can't evaluate exact size of the router's queue, it is very useful for the congestion control. If the virtual queue grows during constant data transmission it means that link is overloaded and speed should decrease, because more and more packets are not delivered. Besides this fact if the virtual queue becomes smaller in size it means that additional bandwidth is available. But if sending rate is not constant the situation becomes more complicated. In this case if rate increases the queue's size may also increase in two cases – just as normal reaction on a rate change and if the congestion occurs. That's why the size of virtual network queue couldn't be used as a good congestion indicator. As it was mentioned in previous chapter the network overloading could be detected when packet is lost or sending delay is more than a threshold, but at the same time these approaches have major disadvantages that aren't acceptable for video multicasting due to their inaccuracy.

A possible solution is to use a mixed approach – properties of the virtual queue and congestion indicators such as the delay and the packet loss. Let's look at the so known bandwidth-delay product formula:

$$b = \frac{q}{r} \tag{1}$$

Here $b$ is a maximum possible bandwidth or a sending rate on the link in bytes per second, $q$ is a queue size in bytes and $r$ is a round-trip time in seconds. It's not hard to notice that the bigger size has queue – the more data could be delivered to the destination and that from some moment increasing this value will not affect on the actual amount of delivered data. On the other hand it's possible to predict the queue size if the sending rate increases or decreases. Let $b_1$ and $b_2$ are different sending rates of the same link, then size of the second queue could be represented by the following transformations:

$$\frac{b_1}{b_2} = \frac{q_1 r_2}{r_1 q_2}$$

$$q_2 = \frac{q_1 r_2 b_2}{r_1 b_1} = q_1 c \tag{2}$$

Using this formula the congestion control could be applied as following: every time the sending rate increased $\frac{b_2}{b_1}$ times then the resulting queue should increase no more then by $c = \frac{r_2 b_2}{r_1 b_1}$ times. At each step the congestion control algorithm sets the value:

$$b_{k+1} = 2b_k\,, b_1 = 1 \tag{3}$$

This continues until formula 2 is true. Such approach guarantees that congestion will be found by not more than $\lceil log_2 b_j \rceil$ steps, where $b_j$ is the maximum possible sending rate. Otherwise if the congestion occurred, it could be eliminated if reduce bandwidth to a next value:

$$b_{k+1} = \frac{q_{k+1} r_k}{r_{k+1} q_k b_k} \tag{4}$$

For every next step this algorithm schedules at $r_k$ seconds after, which guarantees that it can adequately notice changings in the network.

But it should be mentioned that formula 1 is not correct for cases where packet loss is a normal behavior of the current link and doesn't related to the network congestion. Also some noise could appear

due to CPU load or other factors while measuring value $r$. That's why the following statement should replace formula 2:

$$1 - \Delta \leq \frac{q_2}{q_1 c} \leq 1 + \Delta \quad (5)$$

The value of $\Delta$ represents a possible error. As for multicasting even small probability of packet loss is considerable and at the same time the noise could be very noticeable, the $\Delta$ should be compromise. For example, $\Delta = 0.15$ could be such value for the wi-fi link with moderate signal. More accurate value could be achieved by a direct measuring of packet loss probability that could be done by using multicast congestion control described in the next section.

Also it's possible to use a LEDBAT delay measurement congestion control [2] together with the proposed algorithm. Such approach increases efficiency of bandwidth utilization. Reducing sending delay is not described here, because it's beyond the scope of this article.

The main difference between proposed algorithm and the TCP\IP congestion control is that on the one hand it rapidly increases the bandwidth and on the other hand it smoothly decreases it if the network is overloaded. Besides these properties it's also designed for multicasting, as each new packet shouldn't wait, until it will be sent. Instead it transmits immediately, which guarantees low delivery delay.

## Multicasting congestion control

While the local congestion control algorithm is quite efficient, it could be improved by using information from multiple sources. A video stream has a property that is called a bitrate $w$ – it's a speed of link in bytes per second. It could be constant or variable. Nevertheless it has some average value during all streaming that is known before actual start. Some multicasting peer-to-peer solutions like ChunkySpread [9] and Tailcast [10] uses a tree topology for data delivery. It's very useful to divide stream into $j$ substreams, because it means that sender can transmit data to several receivers with different speed. For example, packets with identifiers $1, 1+j, 1+2j, \ldots$ belong to the substream #1, packets with identifiers $2, 2+j, 2+2j, \ldots$ belong to the substream #2 and so on. In this case each substream has bitrate equal to $\frac{w}{j}$.

In peer-to-peer video streaming networks one of the most important task is to use as much as possible of available bandwidth of each participant, because in any type of the swarm (mesh or tree) the main index is an end-user receiving delay. It means the difference between a time when packet produced and delivered to the destination. If a video data divided into many substreams it's possible to use nearly all available bandwidth of the peer. But it's also important to transmit data to the same peers during living in the swarm, because effect of "changing hands" needs additional time that impacts on the QoE (Quality of Experience) for the end-user. In congestion control algorithms, that are proposed in existing works, it's nearly impossible, but using the algorithm proposed here it's an easy task for any peer-to-peer tree topology.

For this the virtual queue idea and controlling algorithm is extended. Instead of local congestion control the peer can choose its sending rate according to information received from several destinations at the same time. In such systems the bottleneck is at the sending side, because peers that haven't enough receiving capabilities leave the network soon due to inability to watch the video stream. That's why a global approach that combines ideas described in previous chapter and retrieving information from different sources may be used that could be described as an algorithm that could be set by the next methods:

```
function GetOldestFreeNeighbor(N,s)
begin
  for i:=1 to |N| do
   for j:=1 to s do
     begin
       if N[i].substreams[j] then
         return pair(i,j);
     end
  return null;
end

function GetYoungestBusyNeighbor(N,s)
begin
 for i:=|N| to 1 do
  for j:=s to 1 do
    begin
      if N[i].substreams[j] then
        return pair(i,j);
    end
  return null;
end

event OnBandwidthIncreased(N,b,s)
begin
 for i:=1 to b do
 begin
  p:=GetOldestFreeNeighbor(N,s);
```

```
   if p != null then
     N[p.first].substreams[p.second] = true;
  end
end

event OnBandwidthDecreased(N,b,s)
begin
  for i:=1 to b do
  begin
    p:=GetYoungestBusyNeighbor(N,s);
    if p != null then
     begin
      N[p.first].substreams[p.second] = false;
     end
  end
end

event OnPacketReceived(N,stream_index,packet)
begin
  for i:=1 to |N| do
    if N[i].substreams[stream_index] then
    begin
      Send(N[i],packet);
    end
end
```

Here $N$ is a set of neighbors; $N$ also contains a field "substreams" – a boolean array, where each value indicates if this neighbor should receive packets of provided substream or no; $b$ – number of substreams count that could be increased or decreased during current congestion control. OnBandwidthIncrease is called each time when the additional bandwidth is avaliable that's determined by (5) and the value $b$ is calculated using (3); OnBandwidthDecrease is called each time when the congestion occurs that's also determined by (5) and value $b$ is calculated using (4); OnPacketReceived is called each time when a new packet is generated by a video source or received from another peer.

Combining the local congetsion control algorithm from the previous chapter and the multicasting peer-to-peer algorithm provides an efficient solution. It eliminates necessity of using TCP\IP and at the same time it could be implemented over the UDP protocol. But it also could be implemented as a transport level protocol and embed into operational system.

This algorithm solves the problem of delayed delivery as the sending rate is always chosen according to an avaliable bandwidth and also solves the problem of frequent changings of destination peers that leads to a better quality of expierence for end users.

## Conclusion

In this paper presented a new congestion control algorithm and its implemantion for a tree-based peer-to-peer network. Separately existing algorithms were examined with their benefits and drawbacks. It has shown that on the one hand most of them have problems in slow detecting and reacting on the network congestion that make them impossible to use in peer-to-peer video streaming. And at the same time some useful ideas such as the detection of sending delay could be reused in the algorithm proposed here.

The virtual network queue is a kind of "sending window" that represents a delayed situation on the real router queue and doesn't have limits in size. This approach results to a better congestion detection, because it doesn't depend on the packet loss that happend due to the network environment. It is used for constructing the congestion control algorithm that is based on the idea of comparing current queue size within expectable value. Achieving optimal sending rate and congestion avoidance that is done by the binary search algorithm and the exact overloading estimation leads to a smooth control of the sending rate. Special attention payed to the possible packet loss and noise during estimation - the proposed algorithm is extended with additional error check for considering such behavior.

All in all the multicasting congestion control algorithm is proposed. This solution extends the idea of the local congestion control to a global level, where several sources (neighbors) of congestion are used for the management. Diversity of sources provides more accurate estimation of the network overloading that leads to a better link utilization.

The results could be used in a tree-base peer-to-peer networks and moreover it guarantees that every packet will be deliverd in a shortest period of time. It leads to a better quality of expierience for end users and makes possible to significantly reduce a delivery delay in such systems. It was specially designed for a Tailcast topology that has been implemented by the author before. However, exact performance evaluation should be additionaly done and it's a target for the future researches.

**Список використаних джерел**


1. *Grieco L. A., Mascolo S.* Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control //ACM SIGCOMM Computer Communication Review. – 2004. – Т. 34. – №. 2. – С. 25-38.
2. *Rossi D. et al.* LEDBAT: the new BitTorrent congestion control protocol //Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on. – IEEE, 2010. – С. 1-6.
3. *Padhye J., Widmer J.* TCP friendly rate control (TFRC): Protocol specification. – 2003.
4. *Kohler E., Handley M., Floyd S.* Designing DCCP: Congestion control without reliability //ACM SIGCOMM Computer Communication Review. – ACM, 2006. – Т. 36. – №. 4. – С. 27-38.
5. *Winstein K. et al.* Stochastic forecasts achieve high throughput and low delay over cellular networks //Proc. USENIX NSDI. – 2013. – Т. 13.
6. *Floyd S., Jacobson V.* Random early detection gateways for congestion avoidance //Networking, IEEE/ACM Transactions on. – 1993. – Т. 1. – №. 4. – С. 397-413.
7. *Feng W. et al.* BLUE: A new class of active queue management algorithms //Ann Arbor. – 1999. – Т. 1001. – С. 48105.
8. *Brandauer C. et al.* Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic //Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on. – IEEE, 2001. – С. 122-129.
9. *Venkataraman V., Yoshida K., Francis P.* Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast //Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on. – IEEE, 2006. – С. 2-11.
10. *Hordiichuk O.* Tailcast–A Distributed Multicast System with Low End-User Delays. // *Theoretical and Applied Aspects of Cybernetics*. Proceedings of the 3rd International Scientific Conference of Students and Young Scientists, 2013, С. 279-286.


**References**


1. GRIECO, L. A., MASCOLO, S., (2004) Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. In *ACM SIGCOMM Computer Communication Review*. T. 34. – №. 2. – pp. 25-38.
2. ROSSI, D. et al. (2010) LEDBAT: the new BitTorrent congestion control protocol. In *Computer Communications and Networks (ICCCN),* Proceedings of 19th International Conference on. – IEEE, 2010. – pp. 1-6.
3. PADHYE, J., WIDMER, J. (2003) *TCP friendly rate control (TFRC): Protocol specification.*
4. KOHLER, E., HANDLEY, M., FLOYD S. (2006) Designing DCCP: Congestion control without reliability. In *ACM SIGCOMM Computer Communication Review*. – ACM, T. 36. – №. 4. – pp. 27-38.
5. WINSTEIN, K. et al. (2013) Stochastic forecasts achieve high throughput and low delay over cellular networks. In *Proc. USENIX NSDI*. – p. 13.
6. FLOYD, S., JACOBSON, V. (1993) Random early detection gateways for congestion avoidance. In *Networking, IEEE/ACM Transactions on.* T. 1. – №. 4. – pp. 397-413.
7. FENG, W. et al. (1999) *BLUE: A new class of active queue management algorithms*. Ann Arbor.
8. BRANDAUER, C. et al. (2001) Comparison of tail drop and active queue management performance for bulk-data and web-like internet traffic. In *Computers and Communications*. Proceedings. – pp. 122-129.
9. VENKATARAMAN, V., YOSHIDA, K., FRANCIS, P. (2006) Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In *Network Protocols*,. ICNP'06. Proceedings of the 2006. – pp. 2-11.
10. HORDIICHUK O. (2013), Tailcast–A Distributed Multicast System with Low End-User Delays. In *Theoretical and Applied Aspects of Cybernetics*. Proceedings of the 3rd International Scientific Conference of Students and Young Scientists, pp. 279-286.


Надійшла до редколегії 22.04.14