UDC 004.75

O.V. Hordiichuk, O.S. Bychkov

A PEER-TO-PEER TOPOLOGY AND MULTICASTING ALGORITHM WITH GUARANTEED QUALITY OF EXPERIENCE

Peer-to-peer applications such as BitTorrent solved a load problem of file distributing, but unfortunately these approaches are not suitable for video streaming due to a real-time data generation nature, heterogeneous behavior of peers and underlying network. The main challenge is to develop a robust topology structure and a fast dissemination algorithm that guarantees QoE (Quailty of Experience) for end-users. This paper presents a simple, but efficient and completely distributed topology constructing and data transmission algorithm that is called Tailcast. It is based on an idea of building tailed tree topology, which guarantees low stretch and reliability of the network. A delay penalty due to a peer churn doesn't depend on a network size in the peer-to-peer system proposed in this paper and the dissemination algorithm provides fast video data transmission compared to existing solutions. Proposed system implemented using WebRTC protocol stack and could be executed in modern browsers. Achieved results demonstrate robustness and efficiency of the system.

Introduction

Today most of Internet traffic is a video. It takes over 57 % of all data transmitted in the Internet and this amount will increase up to 69 % by 2017 [1]. With increased amount of disseminated data, cost of hardware and software maintenance also grows. Nowadays only several companies can afford streaming over millions of simultaneously watching users. Big sized screens and video of ultra-high definition (so known UHDTV) make this problem more considerable. That is why this situation challenges lots of researches from the whole world to find new approaches that will solve big load problems. The most trivial way of increasing overall performance is working on network hardware technology improvements such as IP multicasting. But unfortunately adopting this technology is nearly impossible nowadays as it also requires replacing most of the network hardware that serve the Internet.

While it is hard to solve this problem on hardware level, it is still possible to optimize video traffic using peer-to-peer networks. They provide ability to control network traffic on a software level that eliminates necessity of hardware replacement. Another important advantage is nearly unlimited network resource due to a simple fact that most of Internet connections are symmetric, which means that upload and download speeds are equal and most users in the network can contribute at least equal amount of bandwidth to its demand. Moreover this effect could be increased if it is optimized with information about local peers. However peer-to-peer networks lacks of stability, because they are heterogeneous. It means that every peer can join or leave the network in an unpredictable manner. Such behavior of peers performs changes on the topology structure and therefore impacts on QoE (Quality of Experience) of other peers.

Although there exist lots of video-streaming solutions, all of them suffer either from big delivery latency, from low robustness of the system or provide good performance only in special environments. This paper proposes a distributed system that consists of the topology constructing and self-repairing algorithm as well as the data dissemination algorithm that is called Tailcast. A main target of this system is minimizing data transmission delay and maximizing reliability of the network topology. This system implemented using JavaScript language and WebRTC protocol stack that is available in modern browsers and makes possible to build more complex video-streaming systems without installing additional software for the end-user. However, the proposed topology as well as the data transmission algorithm could be implemented using custom protocol built over UDP and a congestion control algorithm that is also described in this paper.

Related work

Generally, there are only two possible ways of distributing video data: "pull" and "push" approaches. In a case of "pull" strategy every peer announces information about available chunks to its neighbors and in a result they can request new chunks by sending appropriate command. The main advantage of this approach is a fact that peers can be united into a topology of any type. However, this also double dissemination delay as every chunk should be announced before it could be requested. Moreover, for avoiding inefficient bandwidth utilization, buffer-maps are distributed every T seconds which increases the upper bound of an overall delay on value kT, where k – is the height of the graph network structure. Unstructured topologies are the most popular approach for video streaming peer-to-peer applications. PRIME [2] is one of possible implementations for such system. Here authors solve problem of content and bandwidth bottleneck using receiver-driven behavior of peers. A delay problem is not directly addressed in this paper, but system provides a tradeoff between performance and quality of experience. Most of peer-to-peer approaches for video streaming use UDP protocol as it has predictable delivery delay and a size of the chunk is typically equal to a minimum upper bound of the MTU (Maximum Transmission Unit) value. However, in MyMedia [3] system, which is extended for usage in mobile devices, HTTP streaming used with MPEG-DASH standard. Another example is a CoolStreaming [4] where classical "pull" algorithm implemented as well as strategies for recovering after failure events. All of these systems suffer from big delay problems, but at the same time they can survive even a high churn rate due to undirected data dissemination behavior.

In case of "push" approach a sender side considers what data will be delivered as well as its destination point. This significantly decreases transmission delay as redundant operation of available data transmission is omitted. Perhaps the first attempt of using tree-based topologies was Overcast [4], where the "Up/Down" algorithm was used. The key idea is to move each node as far as possible from the root without losing bandwidth performance. Also it stores and updates information about all of its descendants. As a consequence of this approach each peer starts positioning from the root that leads to its overload. Down to the tree load decreases, but the closer node is to the root the more descendants it should serve and from some big value of the network size the topmost nodes will not be able to process new peers. In the Tailcast new node may be connected to a random (any) node in the stream that provides same load distribution among all peers and complete decentralization of the system. Naturally that most of "push" systems form a tree-topology and is known that if remove any element from such topology then all remaining children nodes will become disconnected from the network. For avoiding this problem some approaches try to use hybrid topologies, where mesh is combined with the tree data structure, like it has done in AnyCast [5]. Here in the mesh topology could exist several tree topologies with best multicasting capabilities. The complexity of this operation grows with network size and frequent churn events lead to a poor QoE.

For solving the big latency and the low robustness problems hybrid algorithms such as Prime [6] and mTreebone [7] were proposed. It is the most promising way of creating video-streaming peer-to-peer networks. Here combination of tree and mesh topologies provides a reasonable tradeoff between reliability and performance. In the first approach a random mesh is built and data video stream is distributed using "push" strategy among different subtrees. On the one hand, missing chunks are distributed using "pull" strategy among peers from different subtrees. On the other hand, mTreebone builds a tree from peers with a good reputation, while others form the mesh. Here the reputation value is presence duration of each peer. It is assumed in this work that probability of a peer churn depends on its time presence. As a result this approach provides better (as compared with the mesh-based topologies) end-user latencies. The Tailcast also uses peer's reputation value for constructing topology.

Topology structure

The topology presented in this paper has name "Tailcast" due to a specific chain structure where peers from a head of sequence disseminate data to their neighbors and nodes that located in the tail. When peer joins the network it takes the last place in a chain topology (Figure 1). Therefore, all peers are always sorted by a duration presence in the network starting from the most stable peer and ending by the newest participant. This approach guarantees that stable peers will be always closer to a source of video stream and thus will get better QoE (Quality of Experience) than peers in the tail. It should be mentioned that this topology forms a directed graph, which means that peers are able to disseminate data only to neighbors that are in the right side of the topology if assume that the source is always located in the left side. While the chain topology has easy implementation and maintenance characteristics it also lacks stability and reliability as when even one peer leaves the network it makes one part of it disconnected from another. A solution of this problem implemented using a next approach: every node holds a list L, $|L| = k$ of addresses that contains IP and port information about its predecessors. In case when peer leaves the network the corresponding neighbors erase address information from their list and request parent location from the farthest node in the remaining list. At the same time they also propagate L to a successors, which takes the first value and replaces a record in its own list with an index $k - |L|$. After this the node erases the first value in the L and sends it to the next successor. This operation continues while $|L| \neq 0$ .Value of k should be chosen according to the peer failure probability. The chain will break if all predecessors leave the network simultaneously. If the failure probability of one peer equals to p then the chance of chain breakage will be $P = p^k$. While a big value of k provides better robustness it also introduces a notification overhead, thus a good tradeoff should be chosen. The Tailcast uses $k = 7$ because it provides good robustness even if the node failure chance is equal to 0.5, in this case the list of predecessors will become empty with probability $P = 0.5^7 \approx \approx 0{,}008$. If the chain failure event occurs peer should to reconnect the network via bootstrapping node.

In the Tailcast each node has its own unique ID (identification), which represents its position in the chain and is used for finding new links in the network. A process of finding new links is described in the next section. The source has always ID being equal to 1, its successor being 2 and so on. When a new peer joins it assigns incremented ID of the last node in the chain. When someone leaves the network a successor node decrements its own ID and sends it down to the own successor. The receiver of the message updates its ID according to the predecessor and resends it down to the tree until the last node will be reached.

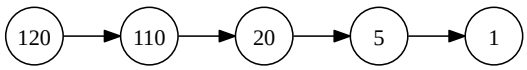

Figure 1. The chain topology. Numbers represent duration (in seconds) of stay in the network. Nodes are always sorted in a descending order

While the chain topology provides good robustness and sorts all nodes by reputation it has propagation delay proportional to the network size, thus it is not suitable for a system with big amount of users. Moreover, each peer has different capacity of a bandwidth and may upload incoming video data to more than one successor that may significantly reduce the latency for nodes that are at the bottom of the chain. While propagation delay of the chain topology has upper bound equal to $|N|$, in a directed n-ary tree topology this value is equal to $\lceil \log_n |N| \rceil$ hops, where N – is a set of nodes. For this purpose the chain topology is extended to contain trees where each node contains l links to other nodes at a distance $2^0, 2^1, 2^2 ... 2^{l-1}$. The distance here is a difference between node IDs (see figure 2). This is very similar to popular DHT (Distributed

Hash Table) systems like Chord [8] and Kademlia [9] where similar approach is used. When node needs to find another node by ID it first looks at its own list of links if it already exists. If there is no such node then it finds the closest node in its list and forwards the request to this node. There will be no more than $\lceil \log_n |N| \rceil$ hops until the searchable node will be found. This approach of storing edges allows a new node to be connected with any (random) other node and find a proper place and links in the chain in a logarithmic time.

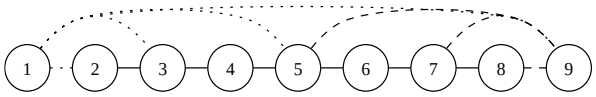

Figure 2. The topology of Tailcast. Numbers represent an ID of the node. Dotted edges represent all links from the node 1 to other nodes and dashed edges represent links from all nodes to the node 9. Tree edges of nodes between 1 and 9 do not present here

Dissemination algorithm and protocol

As it was mentioned in the previous section the topology has directed paths of data dissemination. That is why it is possible to use "push" approach here and avoid additional delay for exchanging available data information as well as their requesting. But this topology doesn't form an acyclic graph that makes impossible data dissemination without duplication. Therefore, an additional mechanism for avoiding loops proposed. A key idea is to provide reasonable performance and loops elimination at the same time. It is known that the video stream could be represented as a continuous file divided into chunks of equal size. We use chunks with 1300 bytes sizes like in BitTorrent's uTP protocol as it seems to be proven tradeoff between real minimal observed MTU (Maximum Transmission Unit) in the Internet and minimal processing overhead. Every chunk has its own ID that represents a time of creation of every chunk created by the source. It could be a real timestamp, but in the Tailcast we use 4-bytes unsigned integers for chunk IDs that are incremented every new entity appearance. Also the Tailcast is built on top of WebRTC protocol stack and uses guaranteed ordered delivery data transmission layer. At the sender side every peer uses these facts for future data dissemination. First of all the peer never sends data to its predecessor. One-direction chunks delivery guarantees better quality for nodes closer to the source. Secondly every node transmits information about the latest known chunk to their nearest neighbors, which helps them to understand current status and make a valid decision for the next chunk delivery. Thirdly the peer transmits data only to those neighbors that have the latest known ID less than its own. These approaches eliminate a possibility for any loops in data dissemination paths and described in figure 3.

It should be mentioned that on the one hand 4 bytes for ID are able to guarantee billions of unique values, but for really continuous streams like TV channels it is possible to notice that a next value after the maximum integer $(2^{32}-1)$ will be 0. This will break original ordered chunk sequence numbers logic. For avoiding this problem we introduce a specific comparison operator that is defined as follows:

$$less(x, y) = (y - x) < (x - y).$$

This operator is applied on computer unsigned number, where minus operator cannot produce negative values. Combining this approach with a fact that at one time a difference between the maximal and minimal values of IDs typically will not be more than 10000 can guarantee that ordering logic will work correct. However, there is still a possibility to attack this network by providing wrong information and therefore break the dissemination that could be solved by introducing reputation peer-to-peer network models, but this is beyond of a topic discussion in this paper.

It is reasonable to notice that peers have different upload capabilities that definitely impact the performance. Moreover they could suddenly change during different reasons like user can start downloading a big

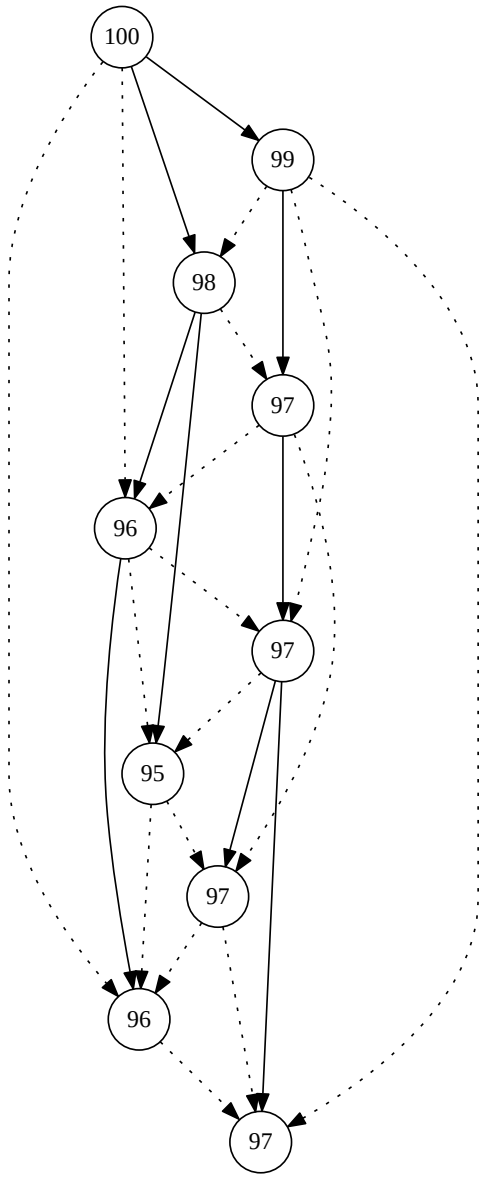

Figure 3. The dissemination algorithm over the Tailcast topology. Numbers the latest known chunk ID to the peer. Data distribution is done by solid edges and dotted edges are used for searching better sources if they will appear

file or a network congestion may occur on ISP level. That is why in this paper assumed that a value *b* (number of links that peer may use to upload the video data without occurring network congestion) known by each peer. Determination of *b* has done by using a congestion control algorithm that was previously specially designed for video multicasting in the Tailcast [10]. This algorithm handles sender's queues that simulate a virtual queue of network hardware, where a transmission bottleneck occurs. A key idea is to keep the virtual queue under a certain load and avoid big overloading that will introduce additional delay. With knowledge of the bandwidth capacity peer uploads incoming video data stream to the nearest peers. While an amount of simultaneously served peers may dynamically vary it doesn't make additional problems for overall performance as a decision for data dissemination chooses clearly for all nodes at any given time.

Implementation and evaluating

Described topology and dissemination algorithm were implemented using JavaScript language and WebRTC protocol stack. While the last technology is still not implemented in all browsers (only Mozilla Firefox and Google Chrome support it), anyway it makes possible to cover more than a half of all browser users in the Internet and avoid installation of additional software. We believe this makes our software more applied for real systems than implementing it as a standalone desktop or mobile application.

The Tailcast was benchmarked using simulated tests on a local host. We used Mac OS X and ipfw tool for simulating network delay and packet loss events. A fake video stream represented as a constant bitrate (2 Mbit) continuous file and. We have measured average and maximum observed delay of peers that are leaves of the tree (the farthest nodes) in the Tailcast network. A size of the swarm is equal to 150 peers with different network environment that represented as a neighborhood size for every peer. Every test has run 10 times with 5 minutes duration (see Table 1).

We have noticed that delay mostly depends on a network delay and peer churn rate rather on a packet loss rate. It could be explained with bandwidth allocation system behavior. If some packets are lost then the congestion control algorithm of the Tailcast will resend them and as it efficiently serves virtual queue lost packets are quickly recovered without significant impact on the performance. At the same time observed values are far from theoretical optimum because of connection establishment time overhead that introduced by the browser and operating system.

Table 1. Test runs

Test runs 1–4				
Neighborhood size	3	3	3	3
Packet loss, %	0	2	2	5
Simulated network delay, ms	0	0	10	10
Observed average delay, ms	12.1	15.2	65.3	78.4
Observed maximal delay, ms	20	20.1	81.2	88.2
Theoretical optimum, ms	4.5	4.5	45	45
Peer churn rate, peers per minute	10	10	10	10
Test runs 5–8				
Neighborhood size	3–5	3–5	3–5	3–5
Packet loss, %	0	2	2	5
Simulated network delay, ms	0	10	15	15
Observed average delay, ms	10.2	45.4	74.3	99.2
Observed maximal delay, ms	14.3	72.3	101	129
Theoretical optimum, ms	3.1–4.5	31–45	46.6 – 68.4	46.6 – 68.4
Peer churn rate, peers per minute	30	30	30	30

An achieved result of implementing the Tailcast topology demonstrates efficiency. However we believe that performance could be improved if the sender side will also consider locality parameter when it makes a decision for data transmission. This will be our main direction for our future researches.

Concusion

In this paper presented approaches for building network topologies and dissemination algorithms for real-time video streaming in peer-to-peer networks. Unlike existing solutions the Tailcast does not suffer from delay problems as here data transmitted with "push" mechanism. At the same a specific topology structure, which have a self-repairing algorithm, makes it really reliable like popular mesh-based topologies with "pull" data distribution models. Also this system guarantees better QoE for those users that are the most stable in the swarm and in a result all free-riding peers are always in the bottom of the topology, while real watchers receive stable video stream.

Except technical advantages the Tailcast also has an ability to run in the web-browser due to a WebRTC protocol stack that is used here. This makes possible to create new kind of video-streaming applications like e-learning systems, TV-channels and video-on-demand services in a scalable way without additional cost for an infrastructure. Results show flexibility, efficiency and robustness of the system.

1. "*Cisco* visual networking index: forecast and methodology, 2012-2017." http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ ns537/ns705/ns827/white_paper_c11-481360.pdf.
2. *Klusch M. et al.* MyMedia: mobile semantic peer-to-peer video search and live streaming // Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. – ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). – 2014. – P. 277–286,
3. *Magharei N. and Rejaie R.* "Prime: Peer-to-peer receiver-driven mesh-based streaming," Transactions on Networking. – 2009. – Vol. 17. – P. 1052–1065.
4. *Venkataraman V., Yoshida K., and Francis P.* "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in Proceedings of the "14th IEEE International Conference on Network Protocols, 2006. ICNP'06", IEEE, 2006. – P. 2–11.
5. *Jannotti J., Gifford D.K., Johnson K.L., Kaashoek M.F., O'Toole J.W., and Jr.* "Overcast: Reliable multicasting with an

overlay network," in Proceedings of the "4th conference on Symposium on Operating System Design & Implementation", USENIX Association Berkeley. – 2000. – P. 97–212.
6. *Wang F., Xiong Y., and Liu J.* "mtreebone: A collaborative tree-mesh over- lay network for multicast video streaming," Parallel and Distributed Systems. – 2010. – Vol. 21. – P. 379–392.
7. *Stoicay I., Morrisz R., Liben-Nowellz D., Kargerz D.R., Kaashoekz M.F., and Dabekz F.* "Chord: A scalable peer-to-peer lookup service for internet application," in Proceedings of the "2001 SIGCOMM", ACM, 2001. – P. 149–160.
8. *Syropoulos A.* "Kademlia: A peer-to-peer information system based on the xor metrics," in IPTPS '01 Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag, 2002. – P. 53–65.
9. *Hordichuk O.* A congestion control algorithm for video multicasting in peer-to-peer networks, Bulletin of Taras Shevchenko National University of Kyiv Series Physics & Mathematics. – 2014. – Vol. 2. – P. 112–117.

Data received 18.03.2015

About authors:

Hordiichuk Oleh Volodymyrovich,
postgraduate,

Bychkov Oleksiy Sergiyovich,
docent,
PhD in physics and mathematics.

Location:

Taras Shevchenko National University of Kyiv, faculty of information technologies, 03022, Kyiv, Ukraine;
Lomonosova st., 81,
E-mail: oleg.gordichuck@gmail.com,
bos.knu@gmail.com