PRIVATE GROUPS IN PEER-TO-PEER NETWORKS

Oleh Hordiichuk, Oleksiy Bychkov

Abstract: *Peer-to-peer architectures are designed for the sharing network, storage and computing resources solving a scalability problem. At the same time they lack capabilities for creating private groups that are necessary for a limited access data exchanging – a common task for e-learning systems. In this paper proposed a new peer-to-peer system that is able to create private groups that is built on top of WebRTC stack and thus is capable to run in modern browsers without additional software installation. It solves problems of authentication, data validity, management control and secure peer discovery in distributed network systems by using private and public keys as well as data exchanging rules that provide same level of data dissemination security as in client-server applications. This system provides ability of creating closed groups of students, where teacher is able to transmit data securely and combine low operational cost of video streaming and file sharing advantages of Tailcast topology. Moreover teacher's capabilities are not limited to only data dissemination process, this application gives additional functions like banning and kicking users as well as a secure access promoting to other participants of the network. While proposed techniques are designed and implemented for the Tailcast topology, it's shown that it is possible to apply these recommendations in most peer-to-peer applications including other video-streaming topologies, distributed hash tables and file sharing.*

Keywords: *peer-to-peer, security, distributed systems, e-learning*

ACM Classification Keywords: *C.2.4 Distributed Systems*

Introduction

Rapid development and success of file sharing systems like BitTorrent motivates researches to apply this idea in other direction and e-learning systems are not an exception. Most of modern e-learning systems are using video materials for knowledge representation that are known to be heavyweight in network distribution meaning. That is why distributing media often is not affordable for universities, when we are talking about more than 1000 of simultaneously watching students. In this case if video stream has the lowest quality and high definition resolution (720p) that is equal to 2Mbit/s, university requires a server with network bandwidth equal to at least 2Gbit/s. At the same time peer-to-peer systems are capable to solve this problem without costly allocated bandwidth channels and use bandwidth of watchers instead. However these applications introduce additional delay that could be equal to more

than 1 minute in such well-known commercial peer-to-peer solutions like SopCast. Such behavior makes real-time video streaming and user interaction impossible. That is why in this paper the Tailcast [Hordiichuk, 2013] topology is used as due to its hybrid-tree network topology structure additional delay is minimized. But at the same the system proposed in this paper could be implemented over other existing network topologies like Kademlia [Maymounkov, 2002], Chord [Stoika, 2001], Prime [Magharei, 2009] and others. The only limitations on the network topology for the proposed system is to have a connected graph network and a ability of peer to make sure in validity of the stream source that do not impact on the structure.

Anyway every peer-to-peer system generally do not have any mechanisms for restricting access of data distributed among peers, so in case of e-learning any student has access to any material that is currently available in the network. In this situation it is possible either to make unauthorized copying of data or to attack the network by distributing invalid data that leads to an unpredictable behavior of the whole system. Also the teacher doesn't have any control over situation in peer-to-peer systems. If there exists chat or file sharing function in such system, it is impossible for teacher to ban or kick peers that are distributing spam messages or files anonymously. Another problem is a providing some level of a management access in a case, when the teacher wants to listen student's answer via webcam or distribute own solution of the task over the whole group. In this case due to a decentralized behavior of the system teacher needs to have an ability to notify all students in the group that one of them has permissions to perform this action. As well as an opposite task of returning access is also complicated without direct notification from a central server in peer-to-peer systems. This leads to a situation where benefits of using peer-to-peer networks are nullified by security problems and that is why it is important to develop the system without these restrictions.

It should be also mentioned that ease of use of educational systems is one of the mandatory features in modern e-learning systems. At the same time most of existing peer-to-peer systems require installation of additional software that increase usage complexity. So for solving this problem the system described in this paper is built on top of WebRTC stack that gives ability to use it only with a modern web-browser. However it is not limited only to this protocol stack and same idea could be applied on any modern peer-to-peer protocol. At the same time it doesn't rely on any browser plugins and could be used as a separate library even in desktop and tablet applications of any vendor. These features increase potential of the solution by making possible to integrate it as a component of a more complex existing educational system without additional limitations. However WebRTC is not a pure sockets implementation and it has some design characteristics like impossibility to establish direct connection without a signaling server due to a protocol security considerations. But at the same time such behavior limits ability of decentralized peer discovery in the network. The system proposed in this paper is also addresses these

problems and try to make process as distributed as possible without decreasing security level by using intermediate peers as signaling servers and certified message exchanging process for removing safety limitations.

Related work

Security is a common problem in peer-to-peer networks that is frequently discussed by researches. Most of approaches are based on trust and reputation models, where historical behaviors and decisions of every peer are used to predict how peer will behave in future. Main idea is to use both own and neighbors reputation observation for calculating trust value that is used for making future communication decision. Reputation systems face a problem such as having an ability to differentiate real feedback from the false one. Another common problem is dealing with a dynamic personality of every participant. For some period of time peer could act as a reliable partner, but after change its behavior and attach the network. Or even more complex type of attacks where malicious peers in addition to previous approaches also communicate with their neighbors using different behavior and thus tangle reputation algorithm. These problems as well as other attacks considered in PeerTrust framework [Xiong, 2004] and SORT [Can, 2013], where used several trust metrics combined with reputation recommendations observed from another peers.

Another approach is to use voting mechanism that helps to figure out secure resources. The more valid peers vote in the network the more exactly malicious peer detection will be. Actually this process is very similar to reputation calculation as votes could be represented as reputation values. However peer-to-peer network is not completely democratic as not all of peers have same count of votes. This happens due to that fact that peers suddenly join and leave the network and newcomers could be malicious with higher probability than old peers. Therefore newcomers must not have more votes than stable ones. An example of such systems are described in VectorTrust [Zhao, 2013] and [Gupta, 2003].

While reputation systems are able to win malicious peers attack in case when they are in majority and provide completely decentralized form of security management, they are not designed to solve problem related to private groups creation. First of all these systems are not completely decentralized as they have presence of the teacher that naturally is a leader of the swarm and thus must have an ability to manage its group. However in this case reputation based systems could consider the teacher as an absolutely truthful source and all other peers can have a mandatory rule for processing commands only from secure neighbors. That does not solve a problem, as at any time there is still non-zero possibility to receive a malicious file or another type of information that is unacceptable in any e-learning system.

Therefore the system described in this paper doesn't rely on reputation security; instead it uses private and public keys for signing and verifying commands from the teacher. This is very similar to Skype authorization and command execution process that is briefly described in [Hoßfeld, 2008]. Here a central server signs a token for the authorized client that is used by other peers for connection identification and validation. Also every time someone joins the network it receives a public key from central server that gives ability to validate messages from other peers. It helps participants to split authorized peers from unauthorized ones. In this paper similar idea is used, the teacher acts as a central server and students as peers. However authorization scheme in messaging protocols like Skype doesn't solve problem of validity data dissemination, secure access promotion among peers as well as attacks from malicious peers. These as well as other problems described and their solution proposed in this paper.

Connection establishment protocol description

In this paper we consider the private grouping as a process of creating peer-to-peer system, where one of the peers acts as a leader of the system (teacher's role) and others as followers (student's role). The leader is responsible for authentication process and must have an ability of secure data dissemination. It means that every follower in the system should recognize generated data from the leader and dismiss data from malicious peers. This behavior could be achieved if the leader and followers adhere following rules during establishing a connection:

1. The follower must be authorized by the leader using login and password or a secret message;
2. The follower and the leader must exchange their public keys;
3. If authorization process is successful then the leader signs and sends a connection token to the follower. Also the follower receives connection information about some amount of existing peers in the network as well as a current timestamp;

The token described in step 3 consists from two parts: unique in system id of the follower that is assigned by the leader and expiration time stored in UTC format. This token may or may not have duration limitation depending on a group type. If private group is open only for a limited amount of time then this field is filled, otherwise it is empty. After keys exchanging and authentication processes are complete the follower can connect to other followers using connection information received in the last step. For accomplishing connection establishment between peers they must exchange their tokens and check their validity. As the token signed by the leader with its private key that other peers don't know it

makes forgery nearly impossible if use long-length keys. And at the same time validity of the token could be easily checked by the leader's public key that every peer already has. In the proposed approach we use a classical RSA scheme with 2048-bit length keys and SHA224 hashing algorithm for the digital signature that provides reasonable level of security, low overhead and high encoding throughput. However depending on application requirements these values could be changed.

In case when the token has an expiration time the receiving side also compares this with current system time converted to the leader's one and consider to disconnect the peer if it is already invalid. The converting is possible due to receiving leader's current timestamp in the step 3. While this time is not precise it is still useful for determination expiration time with accuracy equal to a half round-trip time between the follower and the leader that is usually not more than hundreds of milliseconds. In addition to the previous connection establishment time check the follower schedules an expiration check task that has timeout equal to that time. Such approach guarantees that peers will shuffle off expired neighbors even if they silently live in the network. The whole process of connection establishment described in a figure 1 below:

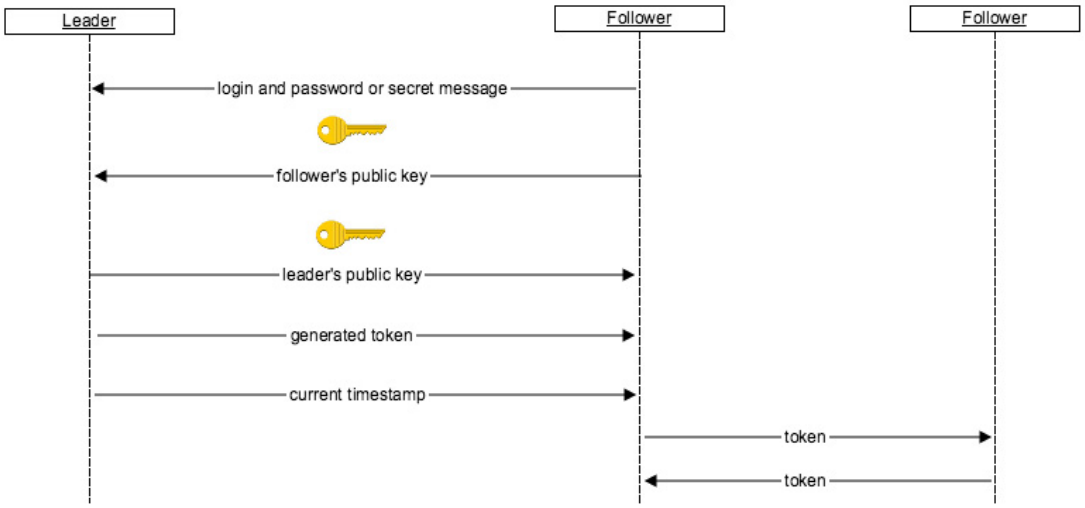

Figure 1. Connection establishment protocol between participants of the network.

Management processes and data dissemination in the private group

As it was previously mentioned proposed system is built on top of Tailcast [Hordiichuk, 2013] topology (figure 2). This topology contains of chain topology, where every peer has a connection with a successor node and at the same time it holds relations with all other peers at distances $2^k, k = \overline{1 \dots n}$, where n is a total amount of users in the network. Such approach is specially designed for low-delay

data dissemination due to ability of every peer reach nearby and far neighbors at the same time. Another important feature of this topology is a clear one-way directed path for data propagation algorithm that gives opportunity to push data without worrying of possible duplications. There are different types of data that could be distributed among peers in such network:

1. Video and audio data including live and on demand streaming;
2. Files of any type;
3. Text messages.

In this topology the leader has a unique ability of a swarm management. The main feature is secure data dissemination. It works as following: every message signed with the private key of the leader and whenever peers receive any data message it can validate it with the public key. It makes this process secure independently of an actual sender of the message. In all cases the source of these data is always the leader or a promoted follower that stands at the same network level in topology as the leader, for further simplicity we call this node “sender”. In newly created group there exists only one sender – the leader, but it can promote any follower by an appropriate command that is distributed among other peers in the network. As it was mentioned in the previous section during connection establishment process between the leader and the follower, the exchange their public keys. So for accomplishing promoting task the leader needs to distribute the public key of that peer that also should be signed by the leader’s private key. After peers receive public keys they can ensure that all messages from the promoted user are valid. In case of the opposite task when the leader wants to refuse in access it can execute a similar process and send a new command that will forfeit promotion. Both of these commands are completely distributed and there is no need to have direct connection between the leader and the promoted follower.

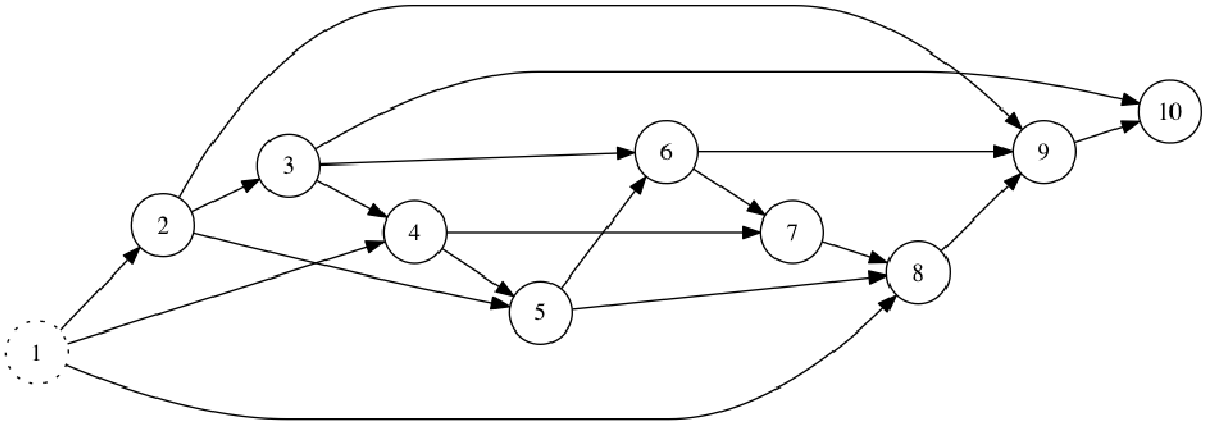

Figure 2. The Tailcast topology. Dotted node represents the leader and solid nodes represent followers.

Another important functions that should have any owner of the private group are banning and kicking users. It is achieved by a similar approach, where the leader sends a signed message to other participants in the network. As it was described in the previous section, during the handshaking process peers also pack their unique ids into the token that help them to identify each other. With this knowledge it also becomes possible to implement ban and kick command. When the follower receives one of these commands and if corresponding id exists in its neighborhood list, it immediately closes connection with that peer. In case when this is the ban command, its receiver also put this id in a ban list and if someone in future will try to connect with the same token it will silently dismiss this proposition and drop the connection. As well as other commands these ones also execute in completely distributed manner.

All information in the Tailcast is distributed using internal stream entity. An elementary information unit is a chunk with a length not more than 1300 bytes that is a typical minimal of MTU (Maximum Transmission Unit) in the Internet. Tailcast data dissemination algorithm guarantees that all chunks will be processed in a correct order and exactly one time. Therefore in this system we pack commands into such chunks. An assurance in order correctness in turn guarantees that all commands will be correctly executed on every peer. However in order to implement distributed chat function we cannot use this approach, as the source of this data could be every peer in the swarm and it is obvious that all of them can't be promoted followers at one time due to the fact that the topology becomes invalid. So for avoiding this problem peers use simple gossip protocol with signing their messages using their private keys. By gossip protocol we mean following conventions between all participants in the network:

1. The peer that is a source of the message generates a random id for it and signs it with own private key. After that peer sends the message to all known neighbors;
2. A receiver of the message proxies it to other peers in its own neighborhood except the sender and put in its history list. In case when there exists another message with the same id in the history list it just ignores it that means both processing and resending avoidance. Every message in history has a time-to-live limitation that is equal to 10 minutes (configurable parameter).
3. Every participant of the network is programmatically limited in its possible sending rate that is equal to 1 message per second (configurable parameter). If the user wants to send with a higher rate than it is alerted with corresponding message about sending rate excess;
4. In case when someone exceeds the sending rate, the receiver side silently drops that connection and does not transmit last message to other peers.

These conventions guarantee that every chat message will be delivered securely and avoid possibility of distributing spam. It is obvious that messages could appear in different order on different computers due to network delay. While it is possible to avoid this problem by using leader's clock information that every

peer knows and delaying messages before displaying them on monitor for previous fetching, it also introduces noticeably delay that significantly impacts on user experience. That is why in our system we display all messages right after they received. Also all gossip protocols have “at least once” message delivery that on the one hand guarantees delivery and on the other introduce transmission overhead at every node that is equal to a number of relations with other peers. However we believe that is a reasonable overhead due to a sending rate limitation that will not impact on system performance comparing with video and file data that is distributed in the network.

Implementation details

While there exists lot of other protocol stacks like BitTorrent’s UTP or even raw UDP sockets that gives ability to implement this system with a minimum overhead and maximum flexibility, all of them need installing additional software on your computer. As it was previously mentioned in the beginning of the paper this system is implemented using WebRTC stack, which makes possible to run a system inside a web page of a browser. This feature is obviously very important for integrating into any modern e-learning system and that is why we have chosen exactly WebRTC and JavaScript as a language. At the same time this technology has limitations. First of all this is still a young technology and currently only the latest versions of Firefox and Chrome browsers support it. Also one of the most important behavior of WebRTC client is bidirectional connection establishing process called SDP (Session Description Protocol) that is impossible without a helper server or node. In our system the leader helps to establish connection with first peers in the system and after that every follower independently discovers the network using neighbors as helpers. This pattern is implemented in P framework (http://ozan.io/p/) that can establish connection using both WebRTC and WebSocket as helper signaling servers and that is why we use it in our system. For WebSocket signaling server case we have implemented a node.js backend application that helps to find the leader of the group.

Also we have faced with a problem that all browsers do not have built-in support of RSA algorithms as well as SHA hashing that makes implementation of this system more difficult. But fortunately for us there exists open-source projects jsrasign (http://kjur.github.io/jsrsasign/) and Javascript Cryptography Toolkit (http://ats.oka.nu/titaniumcore/js/crypto/readme.txt) that contain signing and hashing functionality as well as capabilities for private and public key generation. It should be mentioned that WebRTC protocol provides encrypted message data layer and thus there is no need to additionally encrypt all messages for defending from man in the middle attack.

Conclusion

In this paper proposed a new peer-to-peer system on top of the Tailcast topology that is designed for a secure private group creation that could be integrated to existing e-learning system. Unlike existing approaches of creating secure peer-to-peer networks that rely on peer's reputation, this system uses public key and token exchanging protocol that helps to verify commands from the teacher and establish connections only with those peers that were previously authorized. It guarantees that all data will be delivered only to authorized peers and at the same time malicious peers could not substitute data that provides a certain level of privacy. On top of this protocol different management commands implemented including kicking and banning users as well as granting other peers rights to run these commands. Separately considered function of chat messages dissemination that is based on the gossip protocol with strict sending rate restrictions. Combining these approaches on the one hand guarantee that all messages will be delivered to all recipients and on the other hand it has strong defense from possible spam attacks.

We attempted to implement a system that is secure, distributed, scalable and at the same time can provide reasonable level of privacy. An execution process of all commands is completely distributed that makes this system scalable and usage of digital signage approach provides privacy. Combining these benefits with that fact this system can ran in modern browsers opens the possibility of creating new type e-learning systems.

Bibliography

[Hordiichuk, 2013] O.V. Hordiichuk. 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 — Kyiv: Bukrek, 2013. 279-286.

[Maymounkov, 2002] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." Peer-to-Peer Systems. Springer Berlin Heidelberg, 2002. 53-65.

[Stoica, 2001] Stoica, Ion, et al. "Chord: A scalable peer-to-peer lookup service for internet applications." ACM SIGCOMM Computer Communication Review 31.4 (2001): 149-160.

[Magharei, 2009] Magharei, Nazanin, and Reza Rejaie. "Prime: Peer-to-peer receiver-driven mesh-based streaming." IEEE/ACM Transactions on Networking (TON) 17.4 (2009): 1052-1065.

[Xiong, 2004] Xiong, Li, and Ling Liu. "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities." Knowledge and Data Engineering, IEEE Transactions on 16.7 (2004): 843-857.

[Zhao, 2013] Zhao, Huanyu, and Xiaolin Li. "VectorTrust: trust vector aggregation scheme for trust management in peer-to-peer networks." The Journal of Supercomputing 64.3 (2013): 805-829.

[Can, 2013] Can, Ahmet Burak, and Bharat Bhargava. "Sort: A self-organizing trust model for peer-to-peer systems." Dependable and Secure Computing, IEEE Transactions on 10.1 (2013): 14-27.

[Gupta, 2003] Gupta, Minaxi, Paul Judge, and Mostafa Ammar. "A reputation system for peer-to-peer networks." Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video. ACM, 2003.

[Hoßfeld, 2008] Hoßfeld, Tobias, and Andreas Binzenhöfer. "Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements." Computer Networks 52.3 (2008): 650-666.

Authors' Information

Oleh Hordiichuk *– postgraduate student in Taras Shevchenko National University of Kyiv, faculty of information technologies, Kyiv, Ukraine; e-mail: oleg.gordichuck@gmail.com*

Major Fields of Scientific Research: Peer-to-peer networks, distributed computing, networking and e-learning

Oleksiy Bychkov *– docent, PhD in physics and mathematics, deputy dean of academic affairs in Taras Shevchenko National University of Kyiv, faculty of information technologies, Kyiv, Ukraine; e-mail: bos.knu@gmail.com*

Major Fields of Scientific Research: E-learning, theory of programming, mathematical foundations in information technologies