Annales UMCS Informatica AI XII, 1 (2012) 97–112
DOI: 10.2478/v10065-012-0023-1

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Notary-based self-healing mechanism for centralized peer-to-peer infrastructures

Grzegorz Oryńczak[1*], Zbigniew Kotulski[2†]

[1] *Jagellonian University, Department of Physics, Astronomy and Applied Computer Science Cracow, Poland*
[2] *Institute of Telecommunications, Warsaw University of Technology Warsaw, Poland*

Abstract – Centralized architecture, due to its simplicity, fast and reliable user management mechanism (authorization, authentication and lookup) and O(1) searching capability, is still a preferable choice for many P2P-based services. However, it suffers from a "single point of failure" vulnerability, so networks based on this topology are highly vulnerable to DoS attacks or other blocking attempts. This paper describes a new mechanism that can be used for centralized P2P networks to prevent a P2P service unavailability after central server failure. High security level is obtained by using notary servers which track server public key changes and collect social feedback from users. This allows not only to detect popular attacks (like man-in-the middle) but also to assess whether the Central Server (CS) behaves properly. In the case of central server failure or when server becomes compromised, decentralized Condorcet voting is preformed and new CS is selected. Additionally, by incorporating a reputation mechanism which uses two kinds of scores respectively for providing good service and fair evaluation of other peers, the best candidates for a new Central Server can be chosen. Valuable data which is used to rebuild user database in new CS is stored in the encrypted form in peers and updated during the user-peer authorization process. The decryption key is divided between peers using the threshold secret sharing method.

*grzegorz.orynczak@uj.edu.pl
†zkotulsk@tele.pw.edu.pl

1 Introduction

In recent years, peer-to-peer systems have become more and more popular in many domains like file sharing, voice and video communication (e.g. Skype), streaming content distribution [**1**], as well as distributed storage and computing centers. As opposed to traditional client/server architecture, nodes of the P2P system (peers) act both as a client and a server and sometimes as a relay for other peers in the network. This ability allows to create more sophisticated and reliable services, which by implementing an abstract overlay network and own routing protocols can gain greater control over the data transmitted via the internet. This ability is especially important for building real-time services like that described in paper [**2**] or systems which provide users with online anonymity like those based on the onion routing technique [**3**] Tor [**4**]. High failure resistance is another big advantage of P2P networks over the client-server infrastructure. It means that if a peer in infrastructure fails, other peers can take over its task and the service will maintain operational stability. Also the cost of building and maintaining P2P based services can be significantly smaller than the client-server approach, because system users may automatically become new peers in the P2P network and provide additional resources (like storage, computational power or additional bandwidth) necessary for service to operate efficiently (like in Skype, where selected users are becoming super nodes and are used to transfer other users data). Additionally, self-organization ability of P2P networks significantly shortens the time needed to create functional infrastructure and increase the scalability of the system.

Typically we can distinguish between three main types of P2P topologies:

Centralized structure – it is the simplest and easiest topology to implement. The central server is used for managing resources shared by peers. Each peer has to actively inform this server about its resources, so they can be indexed in central database. This approach gives an efficient and reliable mechanism for content searching and, after certain resource has been localized, peers can directly exchange content without central server interaction. The central server is also used for user authentication, authorization, and as a user lookup service (an important element of any IP telephony service). The biggest drawback of this architecture is a "single point of failure" vulnerability, so networks based on this topology are highly vulnerable to DoS attacks or other blocking attempts. However, due to its simplicity as well as fast and reliable user management mechanism, it is still a preferable choice for many P2P based services (in example Skype using a centralized login server [**5**]).

Decentralized and unstructured – in this kind of architecture, all peers are treated equally and the network has an unstructured, random mesh topology. Services based on this topology have very good scalability and fault tolerance level, and are much more resistant to blocking attempts. However, this structure lacks any efficient search capabilities, so searching is performed by the network flooding technique which results in large amount of signaling-traffic. Analogically to search, also user authorization/authentication and lookup cannot be performed efficiently or sometimes cannot be performed at all.

Decentralized and structured – in this architecture, network topology has a determined structure. This structure is usually contracted using the direct hash table (DHT) and can take various forms like for example a mesh, ring (like Chord [**6**]) or torus [**7**]. DHT provides a search mechanism similar to hash tables, which is used to associate certain resources (for example file ownership) to a certain peer. This topology has many advantages like good scalability, satisfying fault tolerance and reliable search mechanism (typically $O(\log N)$ messages to resolve a search query in the N-nodes network) which makes it a good candidate for file sharing services. However, in many aspects (user managing, fast data lookup) centralized topology is still much more efficient.

A more comprehensive survey of P2P networks can be found in [**8**]. As can be seen, each of those three main topologies has its own advantages and drawbacks. In this paper we present a novel self-healing mechanism that can be used for building systems that will incorporate both: the fast and reliable searching and user managing mechanism appropriate for the centralized P2P systems and high failure tolerance level as well as blocking resistance known from the decentralized infrastructures. Additionally, the presented method is easy to implement in the existing P2P systems with the central server, which will result in significant increase of security, while retaining the previous performance. This objective has been achieved by incorporating three additional mechanisms, responsible respectively for central server misbehaviour detection, decentralized voting used for choosing a new server, and central database rebuilding (using encrypted backups stored in peers and secret sharing technique). The main idea of this project is described in the next section. Sections 3 and 4 are devoted to voting and data rebuilding mechanisms, and Section 5 summarizes the work.

2 Project description

The self-healing mechanism presented in this paper consists of three main stages.

1. Central Server (CS) failure/misbehaviour detecting – in this stage any abnormal behaviours of CS are detected. Besides typical, relatively easy to detect, errors like hardware or network failures, there can also occur other types of misbehaviour, which may lead to service unavailability or theft of valuable data. The second type of errors is sometimes hard to detect. In the method presented in this paper, an additional trusted group of servers called Notary Servers is used for gathering social information about CS behaviour and based on that knowledge collectively provides information about CS level of trustworthiness. If this level is low, a new server is selected.
2. New server selection mechanism – in the case of unavailability/failure of the current Central Server or if it becomes compromised, a new peer is selected to become a new CS. Choosing a new peer to replace the old CS is done by decentralized voting within the P2P network. The additional data from the reputation system is used to determine a list of the best candidates and divide unequally (depending on their

reputation) a number of votes between peers. A whole voting schema is described extensively in the next section.

3. Database rebuilding – after the new CS is selected, it must rebuild the users database (logins, passwords, additional account information) for a service to be operational again. Data necessary to restore the database is stored in peers in the encrypted form, and is updated during the user-peer authorization process. Besides the users' data, other information can also be partially restored, for example in the movie streaming services if peers buffer is sufficiently large, it is possible to gather parts of movie from peers and recreate the whole file in the CS database.

Operation schema for peers is presented in Fig. 1. It describes a course of action in the case of CS failure.

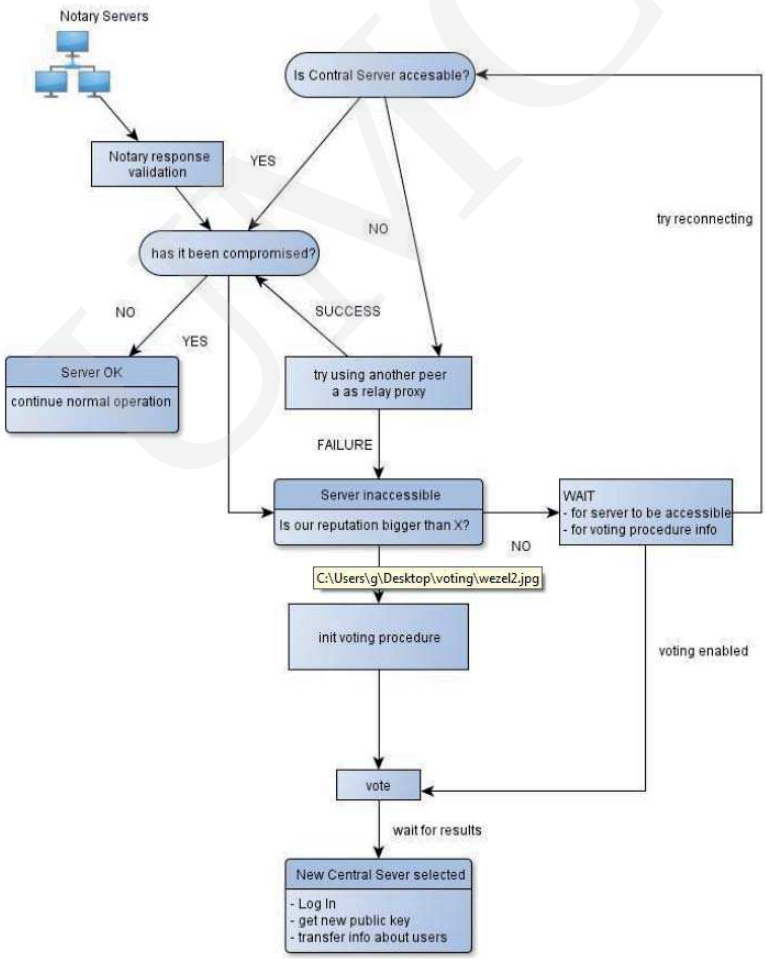


Fig. 1. Self-healing schema – from the peer side.

Depending on the peer reputation score, it can initiate voting for a new CS, or if its score is low, wait for voting to be imitated and a new server selected. If the voting process is initiated unjustifiably (server is working and has a high degree of reliability) voting is cancelled and reputations of initiating peer lowered (to minimize attempts to service destabilization by continuous changing the CS). The described P2P infrastructure uses a centralized reputation system, which is a natural choice in the case of the central server presence. During the normal system operation, each peer is evaluated by other peers which had direct interactions with it. The aspects like uptime, connection quality, response time, reliability and other aspects dependent on the particular service type (in the example buffer size) are evaluated and this evaluation is sent to CS. The reputation score of peer A determined by peer B in time t is calculated according to **[9]**, using the following equation:

$$Rep_t^B(A) = \alpha * Exp_t^B(A) + (1-\alpha)\frac{\sum\limits_{i \in PL(A)} (R_{REC_i} * REC_i(A))}{\sum\limits_{i \in PL(A)} R_{REC_i}}. \tag{1}$$

Where $PL(A)$ is a list of peers directly connected to peer A and $Exp_t^B(A)$ is a subjective experience score of peer B interaction with peer A. Adding a subjective, based on the own experience component to a reputation score is useful, because it takes into account the aspects such as difference between links quality, however, centralized reputation manager calculates reputation scores in an objective way (with α factor set to 0). So, in the first step every peer obtains exactly the same information about reputation scores of other peers, and then subjective components can be added (usually with $\alpha > 0.5$). To minimize the impact of dishonest peers on reputation scores, the additional recommendation reputation (R_{REC_i}) is incorporated. If the recommendation values (REC_i) given by a certain peer are strongly different fromthose given by other peers, this peer will be classified as dishonest and its R_{REC} will be lowered. This may ultimately result in not taking those peers into account while calculating new *Rep*. In the case of P2P systems that already use the reputation mechanism for links quality evaluation, those scores can be transformed into a peer score.

The flow diagram for users is presented in Fig. 2. As shown in the diagram, the greatest emphasis is put on security aspects.

To provide high connection security level, besides the encrypted connection, two-way authentication must be conducted (to protect against man-in-the-middle attacks, which are easy to conduce for example by using the ARP spoofing technique in the LAN network **[10]**). Typically, server authentication is based either on certificates granted by Public Key Infrastructure (PKI), or on self-signed certificates. However, those techniques have some drawbacks, especially in the P2P networks where the server public key may frequently change, maintaining a PKI is costly and self-signing is valuable for network attacks and exposes users to risks of signing the untrusted server. To minimize the risk of interacting with compromised or malicious CS, we decide to use a group of hosts (Notary Servers) – depending on the implementation, they can be

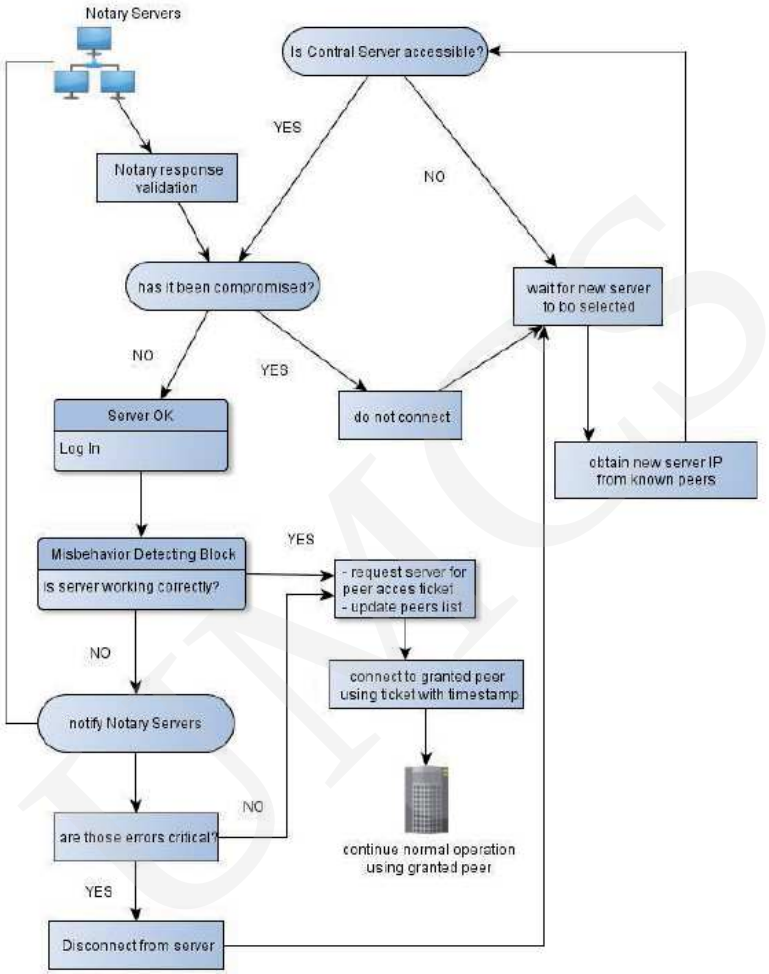


Fig. 2. Flow diagram for the user.

completely independent severs or trusted peers from our infrastructure. One of the main tasks of those notaries is to observe changes of Central Sever public key over time and from different locations. Users can download those logs, and compared to the public key actually presented by CS, many typical attacks can be detected. Further implementation details of key monitoring notaries can be found in [**11**]. The second task of the notaries is to gather social information about CS functionality. During the login process, each user automatically checks actual CS score in Notary Servers, and based on this score and its own security requirements decides whether to continue the login process or to disconnect. CS score is calculated as a ratio of negative opinions to all request. Negative opinions about the CS operation are reported manually by clients and the whole mechanism works similarly to the anti-phishing software [**12**].

3 Voting

Voting algorithms are used in many areas, starting with social aspects like political elections or shareholders voting in companies, to many technical problems like analysis of the results obtained from the sensor network [**13**], where sensors can give different, sometimes erroneous outputs and they need to be agreed. They are also widely used in the distributed computer systems, where in order to perform certain distributed transactions, firstly quorum must be obtained [**14**]. Many spacecrafts used by NASA have redundant control systems and voting is used to minimize the impact of malfunctioning components.

In this section we describe how a voting mechanism can be used for selecting a new Central Server. Voting processes can be categorized in many ways, for example by using a voting algorithm (describing how votes are integrated and the final result obtained), by a number of rounds necessary to obtain the final result or by the type of the obtained result (it can be exact or inexact).

In our case, the main criteria that a desirable voting mechanism should fulfill are:

- Decentralization – because of the lack of any central processing unit, each peer should calculate voting decision independently, due to the P2P network infrastructure, results will be inexact, but it is important that the vast majority of voters should choose the same candidate.
- Speed – a winner should be selected in a single round.
- Fault-tolerance – a relatively small number of malicious or faulty peers should not affect the final result.
- Unevenness – peers with good opinion and bigger seniority should get greater impact on the result than new peers.
- Reliability – a peer selected for the Central Server should be optimal in terms of overall infrastructure performance, should not favour one group of peers and act as a bottleneck for others.

It should be noted that a voting process is the most critical moment in the whole self-healing mechanism, because any mistakes in selecting a candidate for the central server can lead to the whole P2P infrastructure malfunction or/and create serious security problems. In extreme cases the whole infrastructure may be taken over by cooperating malicious peers - for example an attacker creates a large number of entities and tries to gain large, disproportional influence on the infrastructure (like in the case of Sybil attack [**13**] on the reputation systems). Fortunately, in our case, even if infrastructure is taken over - by selecting a hostile peer for the Central Server, any deceitful behaviour of this new server will be detected by users and Notary Servers will be informed, which will result in repeating the voting process and selecting another Central Server. However, this safety mechanism does not eliminate the possibility of stealing valuable data (like information about registered users), so it is important to conduct the voting process in a way that minimizes any possibility of selecting a defective or deceitful peer. Below we describe the most essential parts of a new Central Server selecting scheme.

3.1 Candidates selection and voters weights

Because the P2P infrastructure described in this article already incorporates a reputation system, we choose to make use of reputation scores to improve the quality of voting result. As it was said in the first section, our P2P infrastructure uses the centralized reputation system, so every peer should have exactly the same information about other peers reputation (with accuracy of synchronization time). Such consistency of reputation data allows to reduce the number of candidates only to a small, L size group of those with the best overall reputation score R_{OV} and minimizes the problem of competition among peers (e.g. every peer chooses itself), in current implementation we allow $L = 10\%$ best peers as candidates. Another benefit of the reputation system is ability to distinguish between peers with bigger seniority and good history, and those that are relatively new or proven to be unfair/vicious. However, in this case, unlike in the candidate selecting method, where a candidate overall reputation R_{OV} was taken into account, the key will be ability of peer to honestly assess other peers – which is measured by the recommendation reputation (R_{REC}). Based on R_{REC}, the unequal weight (number of votes) can be assigned to different voters. This procedure, for N voters, produces the weight vector $W = [w_1, w_2, \ldots, w_N]$ where $w_i = f(R_{REC_i})$ is the weight of i voter and $f(R_{REC}) \rightarrow \mathcal{R}$ is the normalization function. In our case, for $f(R_{REC})$ we use a simple threshold function that returns integer in the range $\left[0, \ldots, \frac{(N-1)(N-1)}{2N}\right]$. The number of assigned votes strongly depends on the infrastructure size and distribution of variable R_{REC}. Additionally, it is desirable to prevent an emergence of a dictator (a voter with the weight bigger than the quota).

3.2 Voting algorithm

As it was mentioned previously, the voting process should be fast and due to transmission delays, single round algorithms are preferable. This restriction eliminates standard majority methods (when quote is set for $> 1/2$ or $> 2/3$ of total votes) that are often used for weighted voting. Another basic voting algorithm is based on simple plurality, where entity with the highest number of votes (not necessary with majority) wins. It is simple and fast but the result is sometimes not optimal for our problem. Let us consider a very simple hypothetical P2P network consisting of 23 peers. Let us assume that they can be subdivided into three areas based on their geographical location: areas 1, 2, and 3 with 10, 5, and 8 peers, respectively, like the one presented in Fig. 3.

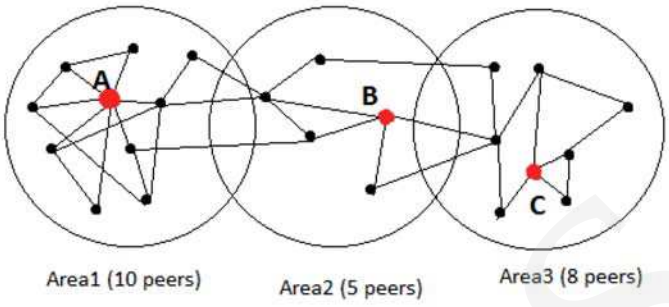


Fig. 3. P2P network subdivided into three areas.

We have three candidates to become the Central Server, marked A,B and C. Additionally, let us assume, that network transmission time delay between candidates and peers in each area can be approximately given as follows:

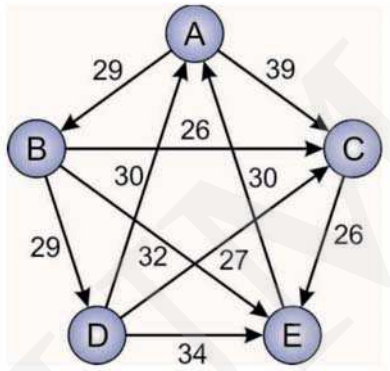
	Area1	Area2	Area3
A	<20ms	<50ms	<100ms
B	<50ms	<20ms	<50ms
C	<100ms	<50ms	<20ms

Access time is one of the main criteria for choosing a candidate by a peer, so, we can assume that in each area peers will be voting for the nearest candidate. As can be noticed, based on simple plurality A will become the new Central Server. Although, from the point of view of the entire infrastructure, this will not be the best option because it will result in large (sometimes unacceptable) delays in communication between the server and peers from Area 3. Based on this simple example, it can be noticed that the algorithms that satisfy the Condorcet criterion (where a final winner known as the Condorcet winner is a candidate that when paired against all other candidates, is preferred by majority) are much better in delivering optimal results. Thanks to this criterion, the selected peer will be the best option for majority of voters (in our example, peer B will be the Condorcet winner) and will fulfill our optimality requirement (in terms of infrastructure performance). Although the Condorcet method has also some drawbacks – it does not take into account problems related to coalitions and there are cases where for a given set of voters the Condorcet winner does not exist, single round algorithms are known, and can be easily applied into the distributed environment. However, the second problem, known as voting paradox (cycles can occur in voters preferences), can be simply overcome by applying another voting algorithm if the Condorcet winner cannot be chosen (the position ranking Borda method is often used in this case). One of the easiest to implement voting methods fulfilling the Condorcet criterion is a Schulz method **[15]** developed by Markus Schulz in 1997. In this method each voter ranks every candidate in order of preference, creating ranked candidates list $v_i = [c_1, \ldots, c_L]$

where for every $i < j$ candidate c_i is preferred over c_j. Based on those lists, the index m is counted, where $m(c_i, c_j)$ is a number of voters preferring c_i over the c_j . Let path P with straight s from candidate c_i to candidate c_j be defined as follows:

$$\begin{aligned} &\forall_{k\in(i,\ldots,j-1)} m(c_k, c_{k+1}) > m(c_{k+1}, c_k) \\ &\forall_{k\in(i,\ldots,j-1)} m(c_k, c_{k+1}) \geq s. \end{aligned} \tag{2}$$

Let $P_s(c_i, c_j) = \max(0, P(c_i, c_j))$ be a strongest path from c_i to c_j. We say that candidate c_i is better than c_j when $P_s(c_i, c_j) > P_s(c_j, c_i)$. And candidate c_w is a potential winner when $\forall_(k \in (1, \ldots, L))P_s(c_w, c_k) > P_s(c_k, c_w)$. Next, the directed graph $G(V, E)$ can be constructed, where $V = (c_1, \ldots, c_L)$ and $E \subseteq (u, v : u, v \in V$ and $0 < m(u, v) > m(v, u))$.



5	ABCDE
7	ACBED
3	ADECB
2	BAEDC
15	BDEAC
8	CDEAB
3	CEADB
3	DACBE
4	EDBAC

Fig. 4. Example of a voting graph with 5 candidates and 50 voters.

For the voting process presented in the form of graph structure (an example presented in Fig. 4), the strongest path $P_s(u, v)$ can be calculated using the shortest path algorithm like for example Floyd-Warshall with the $\theta(n^3)$ computational complexity. The winner will be determined by comparing the results.

3.3 New server selecting schema

For the problem of Central Server selection, we combined the weight assigning procedure with described previously Schulz method, added simple error correcting mechanism and constructed a distributed voting schema. Main steps of this scheme can be described as follows:

1. Candidate selection – each peer p_i, based on the R_OV overall reputation, selects L peers with best reputation score, then creates a vote $v_i = [c_1, \ldots, c_L]$ by ranking these candidates.
2. Broadcast – the votes are broadcast to every peer in the infrastructure.
3. Data collection – for a predetermined period of time, each peer collects broadcast votes, creating a list $V_i = [v_1, \ldots, v_N]$.

4. Input data validation – each peer searches its V_i list for erroneous data and replaces them with the data from its own vote v_i. If the list does not contain values from the specific peer, a missing vote is also replaced by its own one.
5. Votes counting:
   a) Index m is counted taking into account the weight of the individual voters:

$$\begin{aligned}
&\text{for } (k = 1;\ k <= N;\ k++)\\
&\quad \text{for } (i = 1;\ i <= L-1;\ i++)\\
&\quad\quad \text{for } (j = i+1;\ j <= L;\ j++)\\
&\quad\quad\quad m(V[k][i], V[k][j]) = m(V[k][i], V[k][j]) + f(R_REP(k));
\end{aligned}$$

   b) Graph G is constructed
   c) Modified Floyd – the Warshall algorithm is applied for determining the strongest path (P_s)

$$\begin{aligned}
&\text{for } (i = 1;\ i <= L;\ j++)\\
&\quad \text{for } (j = 1;\ j <= L;\ j++)\\
&\quad\quad \text{if } (i! = j)\\
&\quad\quad\quad \text{if } (m[i,j] > m[j,i]) p_s[i,j] = m[i,j]\\
&\quad\quad\quad \text{else } p_s[i,j] = 0;\\
&\text{for } (k = 1;\ k <= L;\ k++)\\
&\quad \text{for } (i = 1;\ i <= L;\ i++)\\
&\quad\quad \text{for } (j = 1;\ j <= L;\ j++)\\
&\quad\quad\quad \text{if } ((k! = j)\&\&(i! = j))\\
&\quad\quad\quad\quad p_s[i,j] = \max(p_s[i,j],\ \min(p_s[i,k],\ p_s[k,j]));
\end{aligned}$$

   d) The Condorcet winner is calculated by comparing p_s, if it does not exist, the Borda count method is applied.
6. The new Central Server has been selected, Notary Servers updated and peers can login. If it happens that a peer locally calculates another winner (e.g. due to errors in gathering the broadcaste date, differences in reputation scores or other errors), it can ask Notary Servers or other peers for a correct result.

3.4 Security

Although in the introduced schema we used only the soft-security technique based on reputation, also the hard security method can be easily applied. For example, in the infrastructures with the sparse network, where many peers are not directly connected, votes signing using a public key cryptography should be considered.

4 Data reconstruction

Next to the new Central Server selecting method, infrastructure data reconstruction is the second essential and most important feature of the model presented in this paper. Before the system can carry on its normal operation, the data like user account (login, password, account balance, etc.) needs to be restored. Also, depending on the service type, other data, like search hashes, resource locations, etc. may be necessary to restore. One way of protection against server failures is to create data backups that will be stored on other servers. However, this approach only minimizes the risk of losing information, and creates additional problems related to data synchronization. Also the cost of maintaining dedicated and secure backup servers in several different geographical locations (in the case of network failure in one data center) can be significant. Additionally, in the case of bigger planned attack in the infrastructure, a few backup servers can be also blocked and service will be not available. Finally, backup stores, which had to be configured previously, eliminate one of the biggest advantages of P2P network – ability to self-organize and operate autonomously. The infrastructure models without predefined storage servers are much more resistant to failures, attacks and blocking approaches, so they are more preferable for constructing services that should be characterized by high independency level (in the examplary electronic cash systems like BitCoin [**16**]).

Another approach for minimizing the data losses caused by Central Server failure is to store backup data in ordinary peers. This gives a bigger level of infrastructure flexibility and safety (more potential data sources) but creates additional problems related to data synchronization mechanisms and data security (ordinary peers are potentially more vulnerable to attacks and attempts to steel data).

In this paper we present new methods for storing duplicated users data in peers of P2P network, that can be used in the case of the Central Server failure to rebuild users' database. Our main goals are:

1. security: data stored in peers should be encrypted with a key unknown to any peer, so in the case when even a significant number of peers will become compromised, the attacker should not be able to decrypt the captured data
2. reliable rebuild schema: it must be possible to rebuild users' database without the previous (probably unavailable/failure) Central Server
3. data synchronization problem should be easily solved.

The method consists of two stages:
The first stage is responsible for data securing and distributing among peers. This process takes place during users' logining into the system. When a user wants to have an access to certain service, he previously connects to the Central Server (with secure TLS connections and the server is authenticated) where it is authenticated (using the challenge-response scheme), then a client is authorized. In the next step Central Server selects a peer that will be used for further interaction with a user, generates a ticket

T, and sends it to the user, which can then present it to the peer (for authentication). The login schema is shown in Fig. 5:

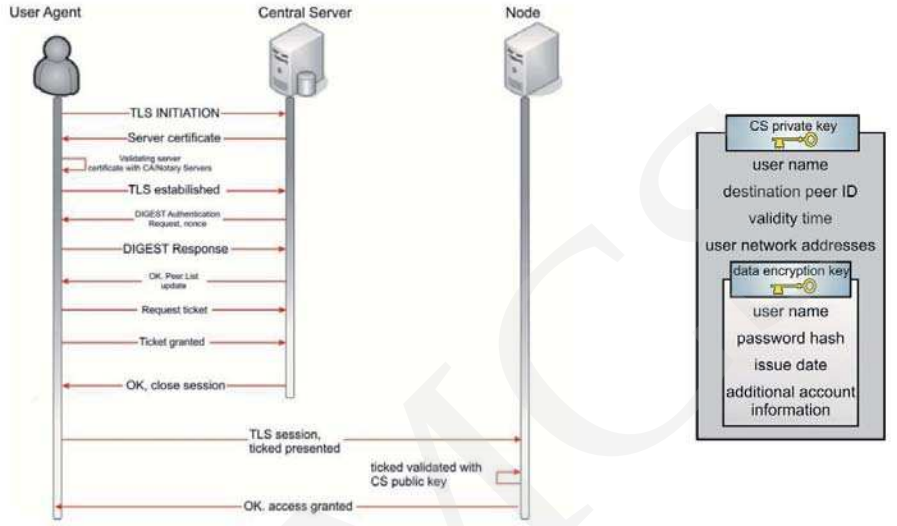


Fig. 5. User login schema and ticket structure.

The ticket granted by the Central Server is encrypted with the server private key K1, so every peer can verify its authenticity (with public key K2). The first part of the data contained in the ticket is similar to that used in Kerberos, so it contains the user's name, destination peer ID, time from (if needed) and until ticked it is valid and a list of network addresses from which the user will intend to use this ticket. Additionally, the second part of the ticket contains the user's valuable data like its login, hash of password, other additional account information and issue date. However, this second part of the ticket is additionally encrypted using another, symmetric key K3, which is known only to the Central Server. Each peer, during the user's authentication, decrypts the first layer of the ticked, and stores its second (still encrypted) part in its database (in conjunction with the user's name and time of receipt). For subsequent logins, if the previously ticked one is not valid and a new one is presented, the stored information is swapped with the new one (for safety reasons, old entry should be retained a for short period of time). So during the time, each peer that had interaction with a certain user, contains the encrypted valuable information about this user's account.

The second stage of the presented method is responsible for K3 key distribution between peers and rebuilding the user's database. For the purpose of rebuilding the user's database, the data stored in peers must be collected and decrypted. To divide K3 key between nodes we use the secret sharing method. Because not all peers may be active at the moment (a few of them may fail, be compromised, or simply log out), the remaining number of peers should be able to reconstruct K3 key, so a threshold schema is needed. Additionally, it should be possible to give more parts of the key to more

reliable peers. Depending on the P2P infrastructure type, size and its stability, suitable threshold should be chosen, in the case of typical P2P network, it can be $t = 80\%n$.

There are a few secret shearing schemas (SSS), that can be used for achieving those goals [**17**]. One of the most popular is the Shamir's schema, which is based on determining the value of the polynomial of degree $(t-1)$ at a point $x_0 = 0$, from given values at t points. For a given (t, n) threshold SSS and k bit K3 key, in the first step the Central Server selects n different, nonzero elements $(x_1, \ldots, x_n)$ from $\mathcal{Z}_p$, where $p > n$ is a prime number. Next, $t-1$ random polynomial coefficients $a_1, \ldots, a_{t-1}$ are selected, and the polynomial $f(x) = K3 + \sum_{j=1}^{t-1} a_j x^j \mathrm{mod} p$ is created. Finally, each i–th peer $1 \leq i \leq n$, obtains a pair (x_i, f_i) where $f_i = K3 + \sum_{j=1}^{t-1} a_j x_i^j \mathrm{mod} p$, using a secure TLS connection. To retrieve the K3 key, the Lagrange interpolation polynomial can be used. After successful voting, the new selected Central Server gathers sheers from peers, then it counts decryption keys K3$= L(0)$ where $L(x) = \sum_{i=1}^{t} f_i \prod_{j=1, i \neq j}^{t} \frac{x - x_j}{x_i - x_j} \mathrm{mod} p$. Next, the data about the user's accounts is gathered from peers' databases, decrypted with K3, and the user's database is restored. Because data is often multiplied, usually the latest entry is selected, although, if necessary, also database status before given time can be partially restored (which can be very useful for example if the previous Central Server generated a few tickets with errors before its malfunction has been detected and Notary Servers informed). However, this method has a small drawback: in the worst case, if the user employs only a small number of peers, and at the moment none of those peers is available, the information about this account will not be restored until certain peers are again available, and the user needs to create a new temporary account. Fortunately, this problem will affect only a small number of least active users.

5 Conclusions and future work

A new self-healing mechanism for the centralized P2P network was presented. This mechanism can be used for building new P2P based systems which will combine the best features from two presently most popular P2P infrastructures: (a) easy and secure user's authorization and management, fast and reliable resource searching and efficient peers load balancing known from the centralized P2P networks with (b) high failure/blocking resistance known from fully decentralized systems. If there are no additional requirements, that only central server can fulfill (like large storage space or additional specialized hardware), also the existing systems based on the centralized infrastructure can adopt this schema and gain higher level of reliability and security without decreasing their performance. High security level is obtained by using Notary Servers which track server public key changes and collect social feedback from users. This allows not only to detect popular attacks but also to assess whether the central server behaves properly. Additionally, by incorporating a reputation mechanism

which uses two kinds of scores respectively for providing good service and fair evaluation of other peers, the best candidates for the new Central Server can be chosen. The distributed voting process, thanks to reputation scoring, is also more resistant to manipulations. Finally, the presented valuable data storing and rebuilding method is secure due to encryption and secret sharing technique, and efficient – data is redistributed automatically during the user's authorization process, so there is no need for additional data distribution and synchronization mechanisms.

However, in the presented schema, there are still some aspects that need to be tuned. Firstly, a secret sharing threshold controlling mechanism should be considered in the infrastructures with a highly varying number of active peers. Secondly, an additional method of dealing with short central server failures should be added. After restoring the server, it should decide whether to make it again a central server, or stay with newly selected and treat the previous one as an ordinary peer – currently only a second approach is used. Also the additional security mechanism must be devised for the situations when the central server becomes compromised, and before this fact is detected and the server blocked, it will try to destabilize the system. For example, server can disseminate false reputation scores and generate tickets with erroneous second part (encrypted user account information), so it will not be possible to use this data to rebuild database after the new central server selection. However, the second problem is currently partially solved by using the additional time delay before swapping data in peers; in the case of the first problem, assuming the distributed reputation mechanism can be used. Also a more comprehensive mechanism for the centralized database rebuilding can be designed, which will for example use data currently stored in peers buffer to restore a bigger block of data on the central server (which can be particularly useful in the case of central server failure of the data streaming service like Video on Demand).

Acknowledgement

Research reported in this paper was supported by the Polish Ministry of Science and Higher Education under Contract No. UMO-2011/01/N/ST6/07387.

References

[1] Hefeeda M., Habib A., Botev B., Xu D., Bhargava B., PROMISE: Peer-To-Peer Media Streaming Using Collectcast, Proceedings of ACM Multimedia (2003): 45.

[2] Oryńczak G., Kotulski Z., Agent based infrastructure for real-time applications, Annales UMCS Informatica 11 (4) (2011): 33.

[3] Goldschlag D., Reed M., Syverson P., Onion routing, Communications of the ACM 42 (2) (1999): 39.

[4] Dingledine R., Mathewson N., Syverson P., Tor: The Second-Generation Onion Router, Proceedings of the 13th USENIX Security Symposium (2004).

[5] Baset S., Schulzrinne H., An analysis of the skype peer-to-peer internet telephony protocol, Technical Report CUCS-039-04, Computer Science Department, Columbia University, New York, NY (2004).

[6] Stoica I., Morris R., Karger D., Kaashoek F., Balakrishnan H., Chord: A scalable peer-to-peer lookup service for Internet applications, Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications SIGCOMM'01 (2001): 149.

[7] Ratnasamy S., Francis P., Handley M., Karp R., Shenker S., A scalable content addressable network, Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications SIGCOMM'01 (2001): 161.

[8] Wang Ch., Li B., Peer-to-peer overlay networks: A survey, Technical Report, Department of Computer Science, HKUST (2003).

[9] Liu J., Issarny V., Enhanced Reputation Mechanism for Mobile ad hoc Networks, Proceeding of Trust Management: Second International Conference iTrust'04, LNCS 2995 (2004): 48.

[10] Callegati F., Cerroni W., Ramilli M., Man-in-the-middle attack to the HTTPS protocol, IEEE Security and Privacy 7 (1) (2009): 78.

[11] Wendlandt D., Andersen D., Perrig A., Perspectives: Improving SSH-style host authentication with multi-path probing, Proceedings of USENIX Annual Technical Conference (2008).

[12] Sheng S., Wardman B., Warner G., Cranor L. F., Hong J., Zhang C., An empirical analysis of phishing blacklists, Sixth Conference on Email and AntiSpam (2009).

[13] Gifford D., Weighted Voting for Replicated Data, Proceedings of Symposium on Operating Systems Principles SOSP'79 (1979): 150.

[14] Jetter O., Dinger J., Hartenstein H., Quantitative analysis of the sybil attack and effective sybil resistance in peer-to-peer systems, Proceedings of IEEE Conference on Communications ICC'10 (2010): 1.

[15] Schulze M., A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method, Social Choice and Welfare 36 (2) (2011): 303.

[16] Nakamoto S., Bitcoin: A peer-to-peer electronic cash system; http://bitcoin.org/bitcoin.pdf.

[17] Karnin E. D., Greene J. W., Hellman M. E., On secret sharing systems, IEEE Transactionson Information Theory 29 (1) (1983): 35.