SeAl: Managing Accesses and Data in Peer-to-Peer Sharing Networks

Konstantin Halachev

Peer-to-Peer Information Systems

Tutor: Peter Triantafillou

1. Introduction

The Peer-to-Peer paradigm is becoming a new standard for architecting distributed applications, with file-sharing systems being by far the most popular among end users. This popularity however leads to a number of challenging, data and resource managing problems. In these systems ,unlike traditional database systems, there is no central authority to manage the storage and computational resources. In the peer-to peer environments each peer manages its own data and computational resources and this leads to a great variance in the behavior of the system.

Peer-to-Peer systems tend to rely on some basic assumptions for the peer behavior on which they base their analisys for effectiveness and efficiency.However recent studies on user behavior showed that the users in peer-to-peer networks tend to behave as selfish as they are allowed to. This leads to a crucial problem of performance, scalability and stability of the system.

The subject of this presentation – SeAl, considers as main task the problem of tackling selfish user behavior.

SeAl is an infrastructure transparently weavable into structured and unstructured P2P sharing networks, which provide the system with possibility to categorize peers and allow a regulated access to the resources depending on their contribution to the society.SeAl manages the service peers receive, depending on their contribution to the society and thus urges peers to be altruistic. This will lead to a better efficiency and overall performance of the underlying P2P system.

2. A high level view of SeAl

SeAl is a software infrastructure which consists of two main distinct layers:

-SAL - SeAl monitoring/accounting layer which monitors behavior of the peers and keeps metadata of the contribution of each peer to the society on which any service the peer received is based.

-SVL – SeAl verification/auditing layer which use cryptographic techniques in order to provide the appropriate security level of the operations of SeAl.

For simplicity reasons we will assume that SeAl works in the context of file-sharing systems even though it is suitable for all other classes of P2P applications.

a. Favors

SeAl counter-selfishness mechanism is based on the “natural” notion of “favors”.
What is favor?
If peer n_1 accesses resource r shared from peer n_2, then we say that peer n_1 owes peer n_2 a favor $f(n_1,n_2,r)$ about resource r.

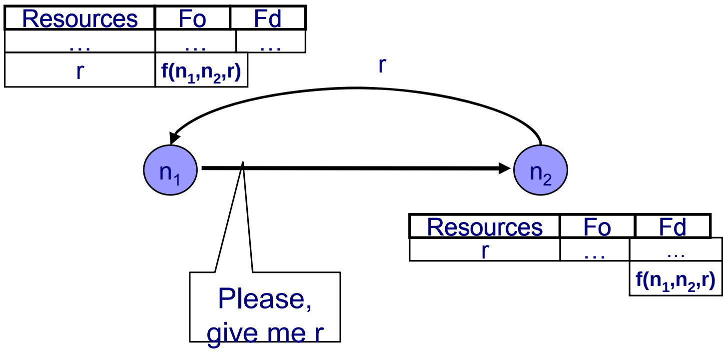


Fig_1 When n_1 accessed resource r shared by n_2, then information about the favor is saved in n_1 and n_2's favors lists

Each node keeps two local lists of information about the favors he participated in.
F_d-the list in which the peer notes the favors he has done,
F_o-the list in which the peer notes the favors he owes to other peers.
Ideally we consider that the perfect load-balancing of a P2P system will be if all peers have accesses the same amount of resources on the network with the amount they contributed to other peers i.e. the equilibrium of the system is when $n_i.F_d=n_i.F_o$ for each peer n_i.
With this observation in mind we will define a selfishness/altruism of a peer as a function of its F_d and F_o lists. We define A to be the altruism/selfishness value for a peer and for A we use either the formula $|F_d \backslash F_o|$ or $|F_d|-|F_o|$.
The higher the value the more altruists the user is but the perfect condition of the system is when the variance of this value given the values of all peer is the least. This will mean that the system reaches a state of load-balancing in which the demands of the system are fulfilled by all peers.

b. Basic notation and infrastructure

Independently of the underlying system SAL deploys a Distributed Hash Table (DHT) overlay of its own to store SeAl specific metadata. If the underlying system already uses a DHT then SeAl can use this DHT for its specific purposes instead of deploying another DHT in the system.
Every node in SeAl has a unique public/private key pair $\{k_p,k_s\}$.This pair is created for each user before his first registering to the system.
The public key for each node is accessible for every other node and when hashed is used as node ID.Which implies that a specific pair of public and private key automatically leads to information of the node past behaviour.For security reasons when a node claims to have a specific key pair upon his initial joining the system he is asked for a verification of the key pair by decrypting some information encrypted with its public key. In this manner peers are prevented from choosing their position in the DHT network.

3. The SeAL monitoring/accounting layer

In this chapter detailed description of the mechanisms used for monitoring and accounting peers' behavior will be given. This layer only tackles with the problem of user selfishness; all other variations of malicious user behavior are handled by SeAl verification layer.

a. Transaction Receipts and favors

Each transaction in SeAl finishes with both sides possessing a digital ”receipt” for the transaction called Transaction Receipt (TR). TR(n_1.id,n_2.id,r.id.t) is a receipt denoting that peer n_1 with id n_1.id has accessed resource r shared by peer n_2 with id n_2.id at time t.

The favors mechanism in SeAl is implemented using TRs.An entry in F_o or F_d is of the form {n_2.id,r.id,t,TR()}.Some of the information is duplicated because TR is digitally signed by both peers and thus we need the second peer id explicitly to verify the TR.

However not every entry in the favors list has the same unit value. The bottleneck of the P2P systems is the network bandwidth,so we assume that each entry in the favors list has a value of |TR.r.size x TR.t/current time|.The calculation of this value also introduces the aging algorithm in SeAl.Each value of each entry weakens with time thus giving a possibility the peer recent behavior to have most influence upon its rating in the system.

b. Favor payback - enforced

Here we introduce the mechanism for favor payback.

The mechanism is based on forwarding requests for resources.

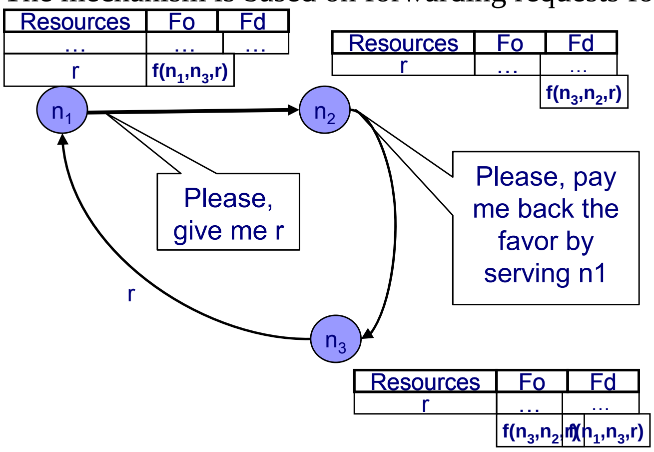
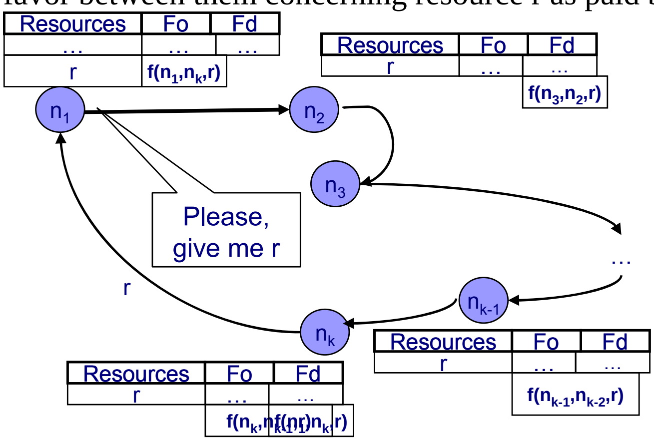
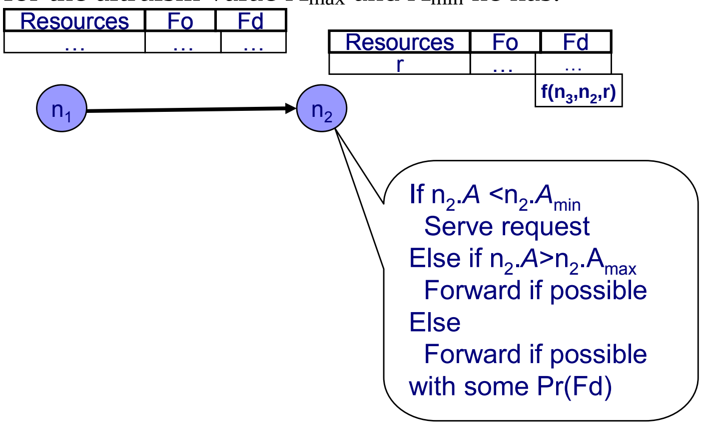


Fig2 Suppose peer n_2 has done peer n_3 a favor for resource r and a new peer n_1 request the same resource r from peer n_2.Then peer n_2 can decide to offer n_3 a chance to pay back the favor by serving n_1 instead of n_2. This is done by n_2 forwarding n_1’s request to n_3 and if n_3 serves it, n_1 marks that he owes a favor to n_3,n_3 marks that he has done a favor to n_1 and both n_2 and n_3 mark the previous favor between them concerning resource r as paid back.



Fig$_3$ This algorithm has a multiple version ,where a chain of forwarding of the request is created but only the last two nodes of the chain mark the favor as paid back because it is considered for the others that the cost of forwarding the request is not equal to paying it back.

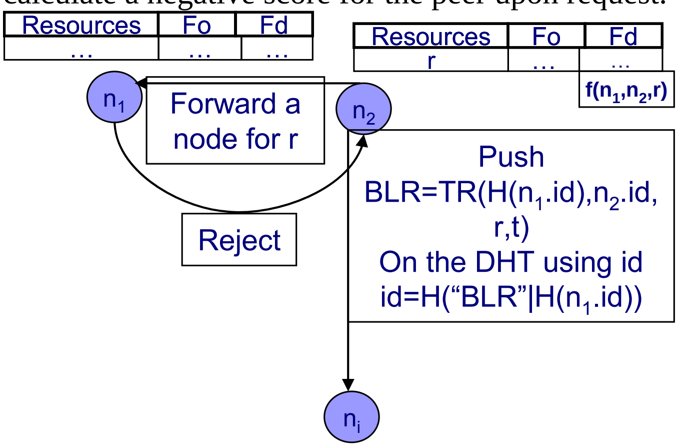
Each peer keeps in track its current selfishness/altruism score in the system- A. Node administrators choose the formula for it :$|F_d|$-$|F_o|$ or $|F_d|\backslash|F_o|$ and they also choose a threshold bounds for A: A_{max} and A_{min}.Each peer upon a request takes the decision of whether to forward it based on its current altruism value and the limits for the altruism value A_{max} and A_{min} he has.



Fig$_4$ Upon a request if peers current altruism value is less than A_{min} he serves the request and thus increases its altruism, if it is greater than A_{max} then he forwards the request if another peer who can handle this request exist. And if the current altruism is in the limits given, then the peer decides its behavior based on some probabilistic method

c. Bad reputation –the black lists

Any deviation from the normal peer behavior is considered selfish and may trigger the blacklisting of the corresponding peer. Blacklisting is publishing a specific black-list request (BLR) entry on the DHT.The BLR is published on the DHT with id=*H*("BLR"|*H*(n_1.id)), where *H* is a hash function and thus the peer that stores the BLR does not know whom does it blacklist. These entries are used to calculate a negative score for the peer upon request.



Fig$_5$. If a peer rejects to serve a favor payback offer, he is a subject of blacklisting and the node whose request was rejected may publish a BLR on the DHT.

d. Request scoring –the white lists

Upon a file request the serving node forms a request score. This request score is formed by a mechanism which takes into account both the peer altruism and selfishness.
The score is formed by the following algorithm:

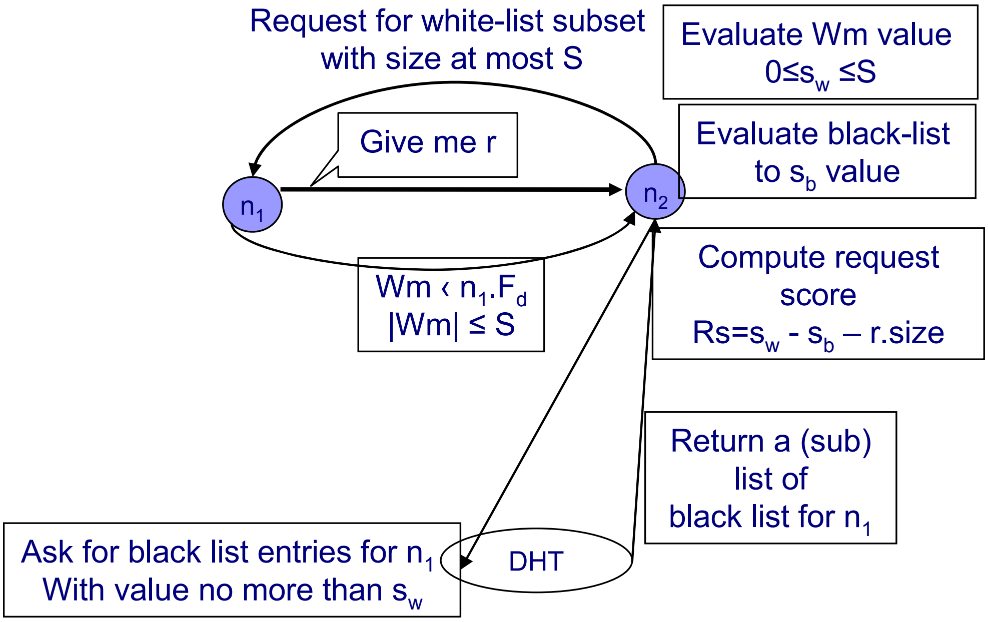


Fig6 Upon request from n_1 for resource r ,n_2 request from n_1 a sub list of $n_1.F_d$ limited by some maximum value S (a limmited list of some favors n_1 has done).Based on the sub list n_1 has sent ,n_2 forms a base positive value of the request s_w corresponding to the value of the white list (no matter how big the list n_1 sends is the maximum value for s_w is S,thus limiting the network overload and malicious user behavior consequences). Then n_2 knowing n_1's id asks the node in the DHT that is responsible for keeping n_1's black-list requests for a sub list of all the black-listings of n_1 with value no more than s_w.Evaluating the received list the negative score s_b for the peer is formed. The total score for the request is formed from the formula $|s_w - s_b – r.size|$.

e. Request serving –the incentives

Why would a peer care about his reputation? Let us overview a simple request serving. When a request comes its score is computed based on the algorithm in the previous part. Then it is stored in a sorted manner in the waiting queue. Based on the decision of the node administrator request with low scores can either be scheduled for processing, allocated limited resources or even rejected. This introduces an important incentatives of user to be altruistic.

f. Debt payback

Peers can regularly check the system for BLR against them. If such exist they can contact the node that blacklisted them and offer to pay back the debt. If all goes well the blacklisted node receives a new TR denoting that it has paid its debt. Then it can either request the node storing the BLR to remove it from the network or wait for the next validation of the BLR to cancel it.

4. SeAl Verification Layer

The SAL layer had a main purpose of tackling with the user selfish behavior, when SVL has to provide the security tools needed for the system operations and tackle all other kinds of misbehavior.

a. Transaction receipt revisited

TRs are the most important object in the SeAl structure so it is natural that misbehaving users will try to attack them. How are TRs protected?
When a transfer finishes both involved nodes receive a transaction receipt which is signed by both the nodes and thus each of them is fully valid authorizing document.Each third party may verify a TR by checking the signatures of the nodes. It is assumed that a signature cannot be forged and because of this a TR cannot be forged too.
Note: Still coluding users can compute TRs which are considered valid from the system.

b. Blacklisting revisited

When computing a request score the serving peer may receive a list of BLR.Then the peer has a task to verify the correctness of the entries in this blacklist request. The theoretical solution is that he checks each entry, but due to the trade-off between extra network accesses and possible incorrect BLR the solution chosen is to verify each of the entries with some probability Pr(v).The algorithm for verification of BLR is briefly described in the following scheme:

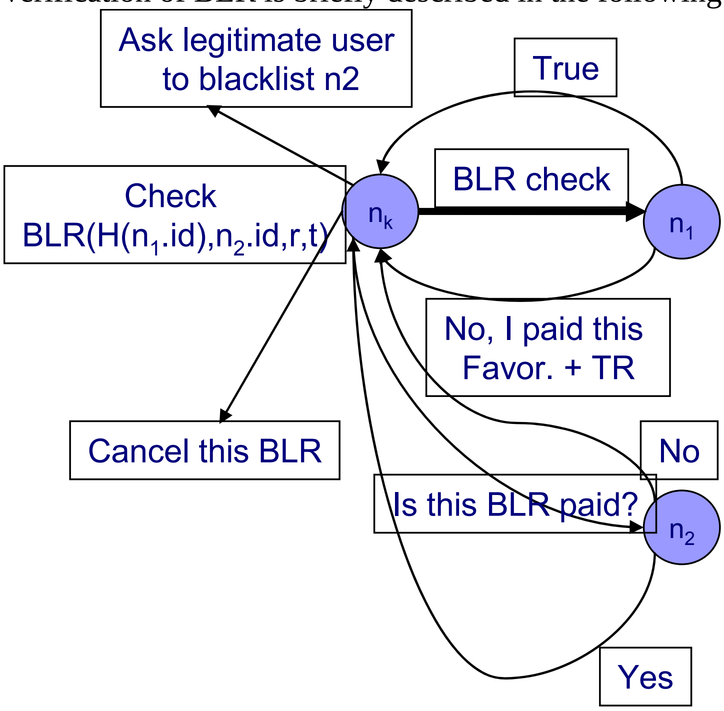


Fig7 Node n_1 was blacklisted by node n_2 and, node n_k is to check this BLR.First he asks if the blacklisted node has any objections for this blacklisting.n_1 at this point has two choices, either he accepts the punishment or he claims he has paid his debt for this BLR. If so he sends a TR that confirms the payback. Then n_k asks n_2 about the status of this BLR.If n_2 confirms that it is canceled then n_k asks the node corresponding for this BLR in the DHT to cancel it. The other case is if n_2 claims

that this debt is not paid, which is considered a misbehaving of n_2 and he is blacklisted.

c. White lists revisited

Similar to the BLR ,each white-list entry is verified with some probability Pr(c).This verification scheme assures us that all the TR used in the system are verified but the overload of network resources is low.

d. File transfer

In this part a simple file transfer algorithm is presented.

Algorithm 1 File Transfer. Algorithm runs on $m.n_{server}$, unlessstated otherwise.

Require:

$send(msg, node\ ID)$: Send msg to node with given ID.

$\mathcal{E}_k(\alpha)$: Encrypt α using key k.

$\mathcal{S}_k(\alpha)$: Sign α using key k.

process(Msg m **)**

Generate k_1, k_2 = random symmetric-cipher keys;

$r_e = \mathcal{E}_{k_1}(r)$; $k'_1 = \mathcal{E}_{k_2}(k_1)$;

$send(\{r_e, k'_1\}, m.n_{client}.id)$;

$m.n_{client}$:

4.1: construct $TR' = \{m.n_{server}.id, m.n_{client}.id, r.id, t\}$;

4.2: $TR'_s = \mathcal{S}_{m.n_{client}.k_s}(TR')$;

4.3: $send(TR'_s, m.n_{server}.id)$;

Verify the signature in TR'_s;

$TR_s = \mathcal{S}_{m.n_{server}.k_s}(TR'_s)$;

$send(\{TR_s, \mathcal{E}_{m.n_{client}.k_p}(k_2), m.n_{client}.id\})$;

$m.n_{client}$: recover k_2 and k_1 and decrypt r;

$F_d.add(TR_s)$; $m.n_{client}$: $F_d.add(TR_s)$;

Fig$_8$.When a request for resource r from the client is received, the server creates two symmetric keys and encrypts r using the first key and the first key using the second. Then these two encrypted resources are sent to the client, but he still needs the second key to decrypt the resource. The client creates an initial draft of the Transaction receipt for this transfer, signs it and sends it to the server. Then the server signs the TR and sends a copy of it to the client together with the key to decrypt the resource.

Note: Still there is a possibility of misbehaviour from the server,because he has a valid TR when the client has nothing but useless bytes.The server can then decide to either just not send the TR,but then this entry will not be valid TR,or black-list the client for this favor .This blacklisting willl be legal,even though the purpose of it is not very clear.

e. SeAl achievements

The above security and verification scheme provides a strong disincentives but still not a complete solution to the common problems of Sybil attack and colluding peers,.

Sybil attack is called the case when a node rejoins the network with new id to obtain a new personality and not be punished for he's previous behavior.

Sybil attack is made undesirable because a peer loses his white list and thus the maximum possible request score increase he can receive is 0,which is the worse possible.
Collusion attack is made undesirable, because no matter how big white lists peer has, their score increase is always limited by the server white list threshold and the size of their blacklists. So it they are malicious users and have huge black-lists they will still get score increase s_w-s_b of 0.
Note:Still when combined the attacks have effect.Which actually means that the colluding peers have efect because Sybil attack does not require neither skill or resources to be launched.However the influence of the Sybil attack by itself is limmited because of the assumption that the newcommers start with the lowest possible request score increase of 0.

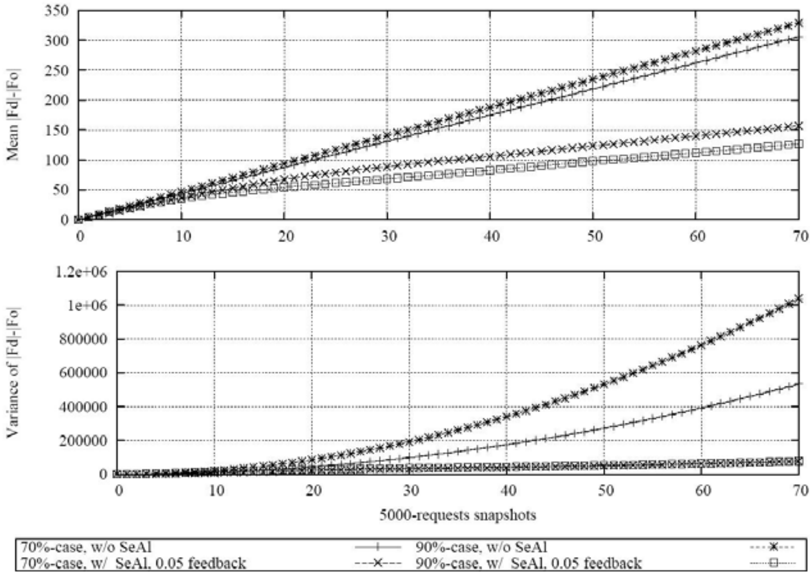
5. Experiments and Performance results

a. Test models setup

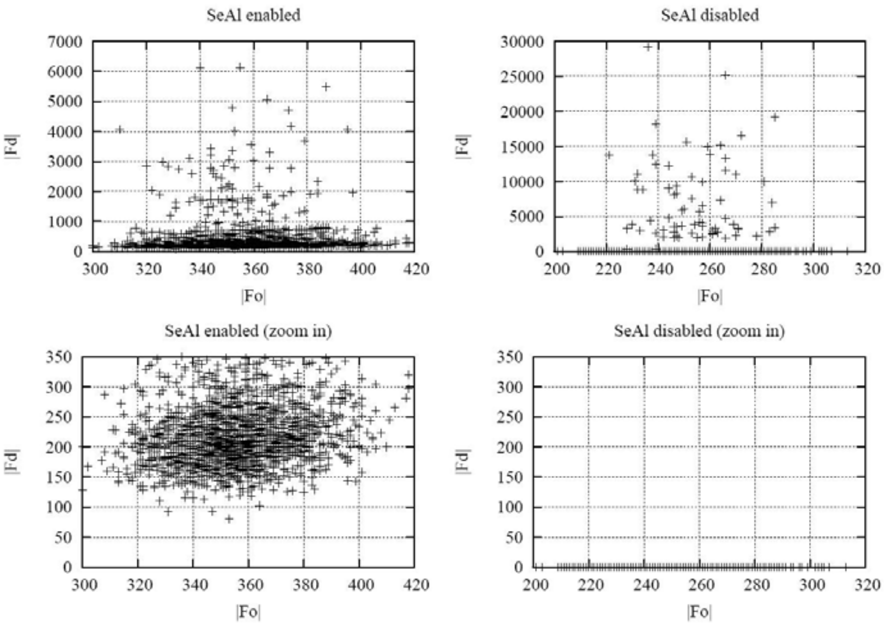
For the experiments the following structure was developed. A file sharing network with 50000 distinct documents of sizes from 3-10MB(average size of 6.5MB).The system has 2048 peers. A simulation of 1000000 request following Poisson distribution, such that every peer will make approximately 5 requests a day of simulated time.
For the peer population two different models were tried:
- 90% freeriders and 10% altruists
- 70% freeriders and 30% altruists

With connections varying from 33,6kbps (modem) – 256kbps (cable) for selfish users and from 256kbps (cable)-2Mbps (T1) for altruists.
Peers compute their altruism scores by the formula |Fd|-|Fo|.They redirect with probabilities 0 when A is below A_{min}, 1 when A is above A_{max} and 0.5 when A is between A_{max} and A_{min}.
Furthermore for describing the user behavior model we have
Pr (Ra) = 0.8 -probability remain altruist,i.e. altruist serve a request
Pr(Rs) = 1 probability remain selfish, i.e. selfish user not serve request.
Pr(Ef)=0.2 per download probability for file erasure
Pr(Ca)=0.1 –per request probability of connection failure
The user behavior is changed by the following mechanism:
When a request is enqueued the client is informed of the expected remaining time until the request is served. If it is more than a fixed threshold value the user drops the request and considers improving his behavior by Sd with probability Pr(Sd)=0.5 SD= 0.05
Note: It is hard to evaluate the objectivity of this user behavior model.It is based on pure heuristics.

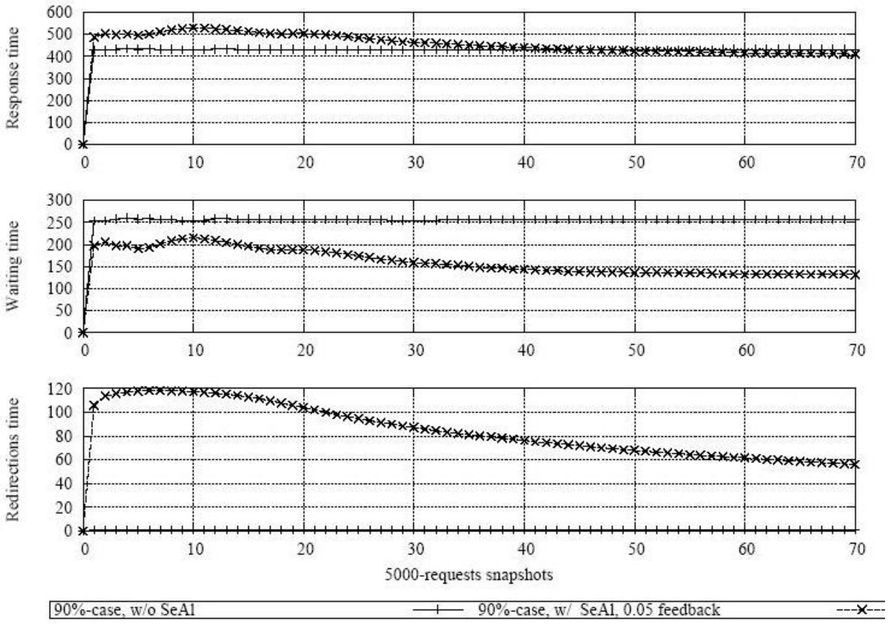
b. **Results and discussions**



In these results we observe that in the SeAl enabled system the mean of the altruism/selfishness function is closer to 0 .As mentioned before the ideal case is considered to be the one when all peers have a selfishness of 0.The important thing is the distribution of the variance where a stable distribution around the mean means that the whole system was balanced and the request were distributed over all peers.



On these result we observe the F_d/F_o graphics for Seal enabled and disabled system. In both systems two clusters are observed, the one of the altruists and the one of the freeriders. The difference of the mean values for the two clusters is really small for the SeAl system and the fact that the cluster of the selfish users is lifted from the base level which means that they mere forced to contribute to the society in order to get served.



(b) 90% selfi sh user population

In the model with 90% freeriders we observe a very interesting fact. The waiting time for the SeAl enabled is slightly lower than the SeAl disabled system even with all the redirections. This model also leads to a better load balancing.

6. Conclusions

SeAl is a system which can be integrated into the current P2P systems. It provides metrics for evaluation and managing selfishness/altruism of the users and offers some incentatives which can be used for regulated access on the base of peer behavior. Still each peer has the freedom to define its own selfishness limits.Network, storage and response time overheads observed in the experiments were significantly small.

SeAl provides a mechanism to limit the influence of Sybil attack and the colluding users problems over the network. With the idea that it can be used as a base for development of wide variety of services in P2P networks.

We have the open problems of initial setting all the threshold values SeAl uses and how are they going to be managed by peers. With SeAl we note down some very interesting but complex as implementation ideas of incentatives for the users.SeAl handles the Sybil attack by giving the newcommers the lowest possible raiting increase score, but even given the complexity of the system, colluding users will still be able to reach a maximum score increase with no significant efforts .Still for some more in-depth evaluation we have to wait until SeAl is tested within a real system with its variety of unpredictable users.