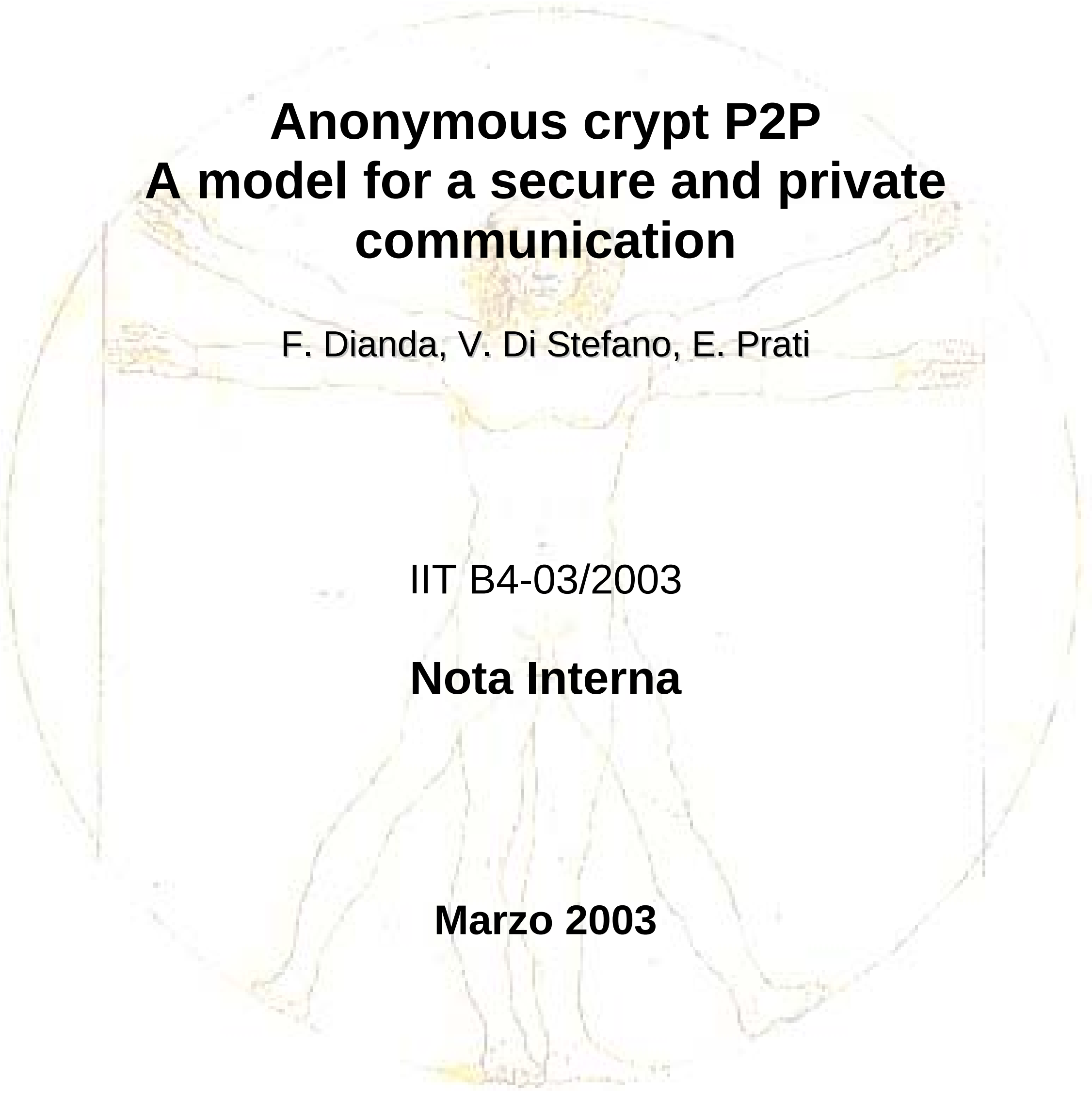
Consiglio Nazionale delle Ricerche



Anonymous crypt P2P
A model for a secure and private communication

F. Dianda, V. Di Stefano, E. Prati

IIT B4-03/2003

Nota Interna

Marzo 2003

Istituto di Informatica e Telematica

Summary

1 Introduction

In the recent years there has been a lot of interest in the research of the distributed software agents and in particular, since the late '90, the *de facto* communication model has been indicated as "peer to peer" (p2p). Instead of the canonical client-server model, in the p2p model every communication entity is able to perform the tasks of client (the requester, the consumer) and the tasks of server (the responder, the producer) at the same time.

One of the most popular p2p working fields has been the *sharing of files* between different users; this activity is so widespread to become one of the source of major dispute in the information society: on one hand the copyright holders of such files and on the other the users that prefer to obtain a copy without the licence to save money.

Another interesting area of activity in the p2p area is the *distribution of information*, in the client-server model, the server is generally the only source to obtain a given information, increasing the interest in a particular resource, the server owner has to install and pay for new computing resources (servers, storage, bandwidth...) to maintain an acceptable quality of service. In the p2p environment, every node that obtains a resource is theoretically able to give it to others requesters, without the assistance of the server.

Lastly, we'd wish to spend some words about a niche area of p2p computing that has increased is importance after September 11th, named the *free but secret speech*: many people are still interested in the ability to express their opinions in such way that anyone is able to ban it.

As anyone can easily image, every model of p2p needs to address some basic security functions to avoid a misuse of the systems. This ranges from the basic integrity insurance of the resources from complex trust and no repudiation management of the communications, to the grant of full user anonymity at every network layer.

1.1 Scope and contents

The aim of this work is to contribute to the modelling of a peer-to-peer protocol in order to fill a lack that still remains in the wide panorama of developed model, i.e. .e. a deterministic anonymous and crypt peer-to-peer communication system. This work considers first the most important model confirmed by the diffusion and the reliability for their purposes, presenting an overview that focuses on the main characteristics. Than an analysis of the requirements is done and two different strategies are analysed, building two models for different anonymity and security levels. The two models are discussed and a communication protocol for a minimal user client interface is described. Finally the scalability problem is discussed.

1.2 Knowledge, References and Activities

Discovering and monitoring of architectures: This function will produce the appropriate knowledge basis to obtain and maintain a snapshot of the state of the art of the *running* p2p application, where running means those proposes that has been coded in a computer executable form. Testing of this software and understanding of their pros and contras, certainly present, will be one important activity.

Developing a running code: This function will focus on the developing not only of yet another new p2p software, but where possible to participate to the efforts of an already existed software. This could cover not only coding activity but could, for example, be represented in a debugging activity or a testing activity.

Use of existing p2p software: In order to understand what p2p means, it could be very useful to become daily user of such software. In this way it is possible to find what particular software:

- Is offering
- Is not offering
- Will be able to offer (with some improvements, patch etc this software could offer this and that too)
- Will not be able to offer (this sw will never be able to offer this and that)

2 A brief survey of the p2p models

It is possible to distinguish between two broad classes of problems in the design and analysis of peer-to-peer networks:

Searching and retrieving of data
Communication and message routing

In both the areas there has been a proliferation of proposals very different one from the other, that is not possible to summarize in few pages; we limit our attention to those with major impact in the security area and those that have received popularity.
It is worth while to introduce some nomenclature to deny confusion or misunderstanding: with the peer to peer we refer in " *a class of applications that take advantages of resources put together at the edges of the Internet"*[5]. Where edges of the Internet could be defined as decentralized resources built and accessible by way of the standard tcp - ip network protocols. So to be more concise we could think of peer-to-peer networks as application level networks build on the top of the Internet.
The study of the peer to peer has much in common with the others distributed systems models, so it could be necessary to distinguish which are the main differences between these models.
Mobile agents are very similar in terms of searching and navigation; they could be defined as a computational entity that moves around the network to perform a task on behalf of an owner. They propagate small pieces of software through the network and not requests like peer to peer so they need a strong and complex security model to grant the receivers from malicious code. Every node must also have a good amount of resources because sending code requires more bandwidth than receiving messages.
Push systems has gained a good acceptance at the end of the `90 due to their capabilities to distribute huge amount of data like multimedia contents. Often the paradigm is depicted as broadcast system where exists two figures, the *producers* and the *consumers*; consumers subscribe, often pay for, a particular service and at a given time this service will be sent across the network.
It is important to note that in the models the communication is asymmetric, only the producers are able to send something through the network; moreover the binding between consumers and resources is static and predetermined (first pay then receive).
Lastly it is interesting to note the similar problems that regards distributed databases and peer to peer network: both need to perform the retrieving of data through a different nodes, so they need an efficient method to discover the location of these resources. But in distributed database it is also important to have a consistent management system that could satisfy the requirements of scalability and consistency needed. So very often some sort of central coordination is involved.

2.1 Searching and retrieving

Searching a resource on a network means holding some information about it and querying for a match of this information, which can be expressed in various forms, range from an assertion on some properties (like RDF) to calculated hash values. It possible to summarize this type of metadata as:

Document calculated: metadata obtained from applying an operation on the resource, the final value of this calculus is considered the keyword used to retrieve the information. Two important examples of this are the digesting through one-way hash function and term frequency inverse document frequency (TFID)

Document assigned: metadata associated with a resource by way of a central authority or from an owner. For example a description of the resources or an RDF assertion about this.

	Pro	**Contr**
Document calculated	Easy to calculate and associate Uniqueness (depends from the operation	Semantic not related
Document related	Semantic related	Difficult to manage (owner choice assigned)

Once established a method to mark the resources, it is possible to use this information to perform the queries, but it is also possible to use the results of such queries to maintain ad update the network topology or the routing tables needed.

Semantic routing: the queries are routed according to the meta data contained in the queries themselves, this implies that every network node has to maintain a routing table or a similar data structure that correlates the discovered associations between meta data and the nodes.

Reputation learning: the queries results are dependent from the answers to similar queries. Reputation informations are maintained locally from each peer and are based from the already established interactions between peers, so they are useful only for it and cannot infer anything about the global network.

Query spaces: the idea behind this technique is to limit the number of possible receivers of a legitimate query; the queries can be classified and send to limit number of nodes able to satisfy the query.

Trust metrics: this approach can be subdivided in

Scalar trust metrics: where given a directed graph G, a starting node s, and a target node t, we say that t is trustworthy and so reachable by s if and only if it exists a path between s and t in graph G.

Group trust metrics: where given a directed graph G, and a target node t, the decision about t reachability is based on the performing of some random walks from a set of starting nodes to t. It is worthwhile to note that this is the approach behind the Google search engine.

Query forwarding: queries are passed from node to node if and only if a node is not able to satisfy it: this approach differs from normal message routing because in the latter queries and forwarded from a requestor to a specific, often pre-established destinations.

This is the approach gnutella networks are based on.

Distributed hash table: the position of a resource in the network is determined by applying a function on the resource itself, the obtained value (the metadata to match) are then stored in particular network location. Peers maintain a section of a global data structure and are able to perform the function to calculate the location of the resource.
An example of this approach is the Chord system [14] where the network is modelled as in a circle; resources are indexed by keys of *m* bits applied to the node identifier (for example the IP address). A peer *p* stores all the keys that fall in the interval:

$$(predecessor(p), p]$$

In practice every peer *p* stores a table of successor *S* such that:

$$S = \{successor(p + 2^{k-1})\} \forall k = 1..m$$

In this way searching can be achieved in O(log n), where *n* is the number of nodes of the network, for example if there are 100 nodes in the network it is possible to locate the a resource in 2 hops.

2.2 Communication and message routing

After discovering the location of resource there are various approach to task of obtaining that resource. One of the main concerns is the network topology, or as the peers are organized and interact; it is possible to distinguish three models. This section describes the architecture and the main features of such principal existing p2p models.

2.2.1 Centralized model

The first model that has gained popularity in the p2p area has been the hybrid model, where the peer relationship is relative to the communication but neither the discovery of the peer node nor to the routing tasks necessary to bring together the network topology.

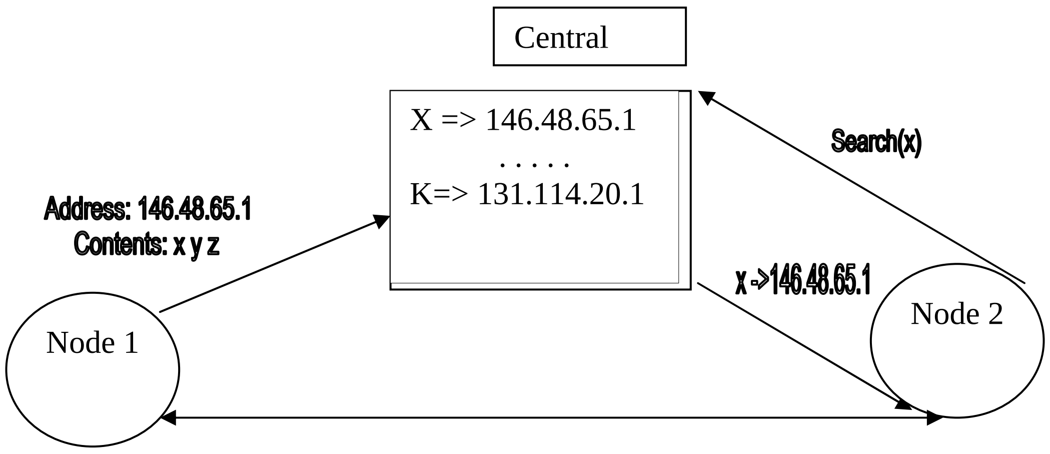


Figure 2.1 Centralized model general architecture

As showed in the figure 2.1, every announce about the joining of the network is announced to a central server whose main task is to record the information that every new peer wishes to share with the others. In this way every search performed by another node could be served looking up in the server repository, like a query on a remote database. After obtaining the search results a peer can contact the other beginning the real communication.

The most famous software that used this model was Napster, a project started from the efforts of a computer science student named Shawn Fanning, which gained a lot of popularity at the end of the ’90. After receiving an huge fame due to the its sharing ability, Napter inc., a company funded by Fannig himself whose business plan was to run the central server, had to shut down after the legal problems caused by the recording and motion associations.

In the next table we summarize the *pros* and the *contrs*

Pro:

- *Easy*: most of the logic needed to perform discovery and routing are on the bulk of the central server
- *Light (for the peers)*: every peer runs is not involved in the global network infrastructure, it doesn’t perform message routing, so it is focused only on its conversations with other peers.
- *Deterministic searching*: due to the presence of a central repository, every search is similar to those performed on a database and is completely independent from network topology

Cons:

- *Scalability:* the central server is the fulcrum; the peers notifies his reach ability to the server once started, so it has to be well-equipped in terms of hardware and bandwidth; an increase of the peers brings to an increase of the costs for the central server. This brings to the need of replicate and redound the database at various network locations with a well-equipped set of clustering and load balancing appliances.
- *Single point of failure:* a successful attack exploited against the central server brings to collapse of the whole network in a short time. System redundancy has proved to be not so easy to achieve and maintain in the client server model.

2.2.2 Pure peer to peer

In this model peers not only perform the tasks needed to communicate with each other, but they are involved in the network topology maintenance. Every peer announces to the neighbours the contents that wishes to share, so every node has a local knowledge of the whole network; in this way every node could be contacted for two different tasks: searching and communication.

Pro:

Completely decentralized: it does not exist a central point of failure

Fault tolerance and scalability: due to its decentralized nature, the networks is hardly to break down, it can pass over failure quite well, but it must be clear, every peer needs to maintain and manage its availability

Contrs:

Topology is unknown: it is not possible to determine a upper bound for the search time, or better searching results are not determistic, because are influenced by the status of the network

Bandwidth hungry: the peers must route the queries on behalf of the peers they are connected to, so resources and in particular bandwidth are used not only for communication, but are directly used to maintain the network up and running.

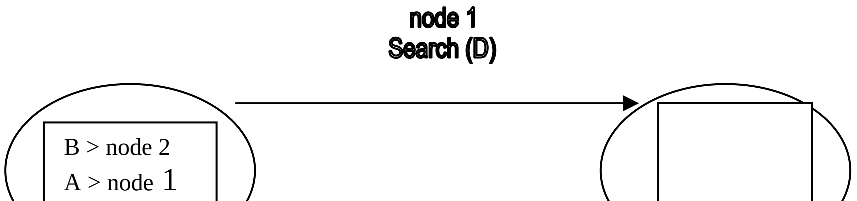


Figure 2.2: The general architecture of the pure peer to peer.

Existing software that uses this approach is Gnutella: in the first version of the Gnutella protocol, a peer (called a servant) broadcasts, to any other servant connected to him, every request that it cannot fulfil. To avoid the flooding of the network, a TTL has been introduced to split the horizon of a request, in this way the number of reachable nodes is given by this relation:

- *n*= number of connected servant
- TTL= number of hops performed so far
- Reachable nodes = n^{TTL}

It is important to note that increasing *n* and TTL:

- Allow to reach a major number of nodes
- Increase exponentially the number of messages exchanged

Due to its broadcast communication and TTL settings, the Gnutella model has proved to be not very scalable; notice that for two times Gnutella network has been out of service because the number of connected user was too high.

2.2.2.1 Freenet

We would spend some words about one the most interesting project currently on going in the p2p area, named Freenet. The key topics in this project are a little different from other p2p projects and could be summarized as:

- Anonymity of producer and of consumer of a given information
- Dynamic routing of the messages
- Adaptive network topology
- Decentralization of all the network functions

It is important to note that there is no guarantee that a particular resource could be permanently stored and retrievable in the network; in short there is no upper bound to the lifetime of a resource.

Otherwise, as showed after, if enough nodes join the network, Freenet could be modelled as a distributed anonymous file system.
It is also important to note that Freenet is built on the top of an existing network stack and *assumes the existence of a secure* – cryptographed - *channel*, so the anonymity concerns are about Freenet messages, not about network usage.

The main concepts in Freenet could be so summarized:

- Resources are indexed and retrieved through keys that are location independent, and assigned through the SHA1 digest value. To be rigorous, there are three types of keys all derived by a very sophisticated user of public key cryptography:

- KSK (Keywork signed key): this type of keys are generated from a short string chosen by the user when inserting a resource in the network, this string is used to generate a key pair whose use in showed in the next figure.

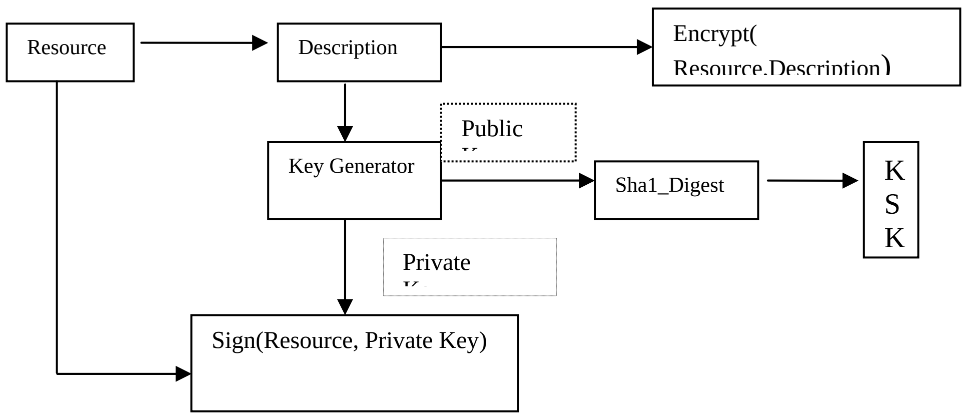


Figure 2.3: Freenet: KSK .

With the signature of the resource with the generated private key it is possible to provide integrity of a resource against its description. Subsequently, the user publishes only the description, a short string easy to remember and to process, this would be the primary key for searching in the Freenet network. It is important to note that descriptions are chosen by the user, so this method is not sufficient to avoid fakes or key squattering: everyone can choose, malicious or not, the same string for two different resources, this is indicated as the flat namespaces' problem.

- *SSK (Signed Subspace Key)*: This type of keys permit the establishment of personal namespace, the main flow is depicted in the next figure.

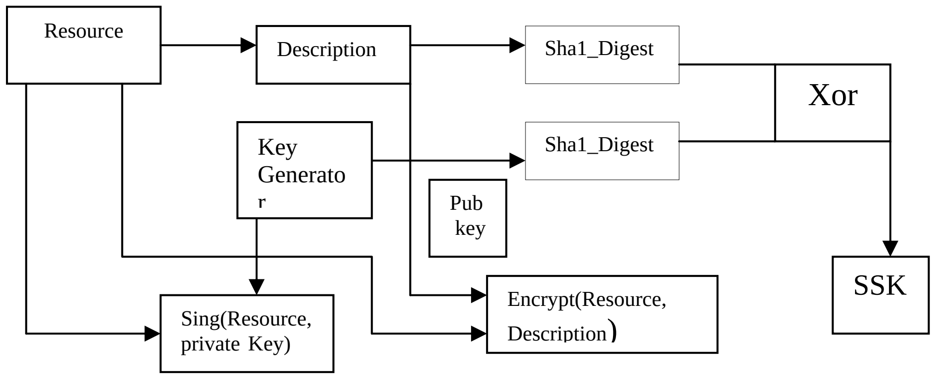


Figure 2.4: Freenet: SSK .

The user distributes the description of the resource and the subspace public key, note that this key is randomly generated, in this way signature of the resource is a less insecure compared to the KSK scheme. The procedure allows simulating a directory like a structure that is managed through the publishing of the personal, randomly generated namespace.

- *CHK (Content Hash Key)*: this type of key is useful for implementing updating and splitting of a given resource.

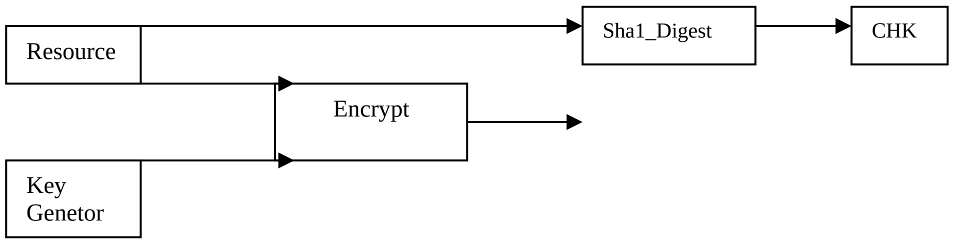


Figure 2.5: Freenet: CHK .

The user publishes the content hash and decryption key; CHK is used in conjunction with SKS to allow the splitting and resuming of resources as described in [7]….

2.2.3 Hierarchical model

In this model there are some special nodes, called super peers, that are able to satisfy particular tasks, such as routing and maintenance of network topology.

2.3 Comparing p2p models

The comparison between the presented models has to deal principally with three aspects, that have to be considered to develop a new model. The aspects concern:

- Centralization of the information
- Privacy

- Probability to find some peer/resource

A hybrid model guarantees the deterministic search of a resource, but depends strongly from a system administration; a pure peer to peer depends from the number of the users and the topology of the resources but is virtually impossible to damage it. There is not a definitive answer: the best depends on the user needs.
The following table resumes the main characteristics of the principal model of peer to peer considered.

Model	**Centralization**	**Privacy**	**Finding a resource**
Centralized	Users announce on server	Depends on server admin	100%
Pure	Local announce to peers	Dependent on the trust on peers	Dependent on the number of users and the topology
Hierarchical	Local announce to peers	Crypt communications	Dependent on the number of users and the topology

Table I: Comparing different p2p models. It is impossible to define the best communication system: it depends on the needs and the requirements of the

3 Diffuse Existing Protocols

3.1 Napster

Napster protocol has never been published; its knowledge is based on a reversing engineering process under the name of openap.

Transport Protocol : TCP for the communication between peers and central servers
Application Protocol format:

Length	Type	Data

- Length: the length of the data portion
- Type: the type of the message
- Data: the data been transferred (generally encoded as pure ASCII text)

Messages and types:

Type	**Sender**	**Description**	**Format**
0	Server	Error message	<message>
2	Client	Login	<nick><passwd><port><client info><link type>
3	Server	Login ack	<user's email>
5	Server	Auto-upgrade	<new version><http-hostname:filename>
6	Client	New user login	<nick><pwd><port><client info><speed><email address>
100	Client	Client notification of a shared file	"<filename>"<md5><size><bitrate><freq.><time>
200	Client	Search request	[Filename contains "artist"]

			Max_Results <max> [Filename contains “song”] [Linespeed <comp><link type> [Bitrate <comp> “bitrate”] [Freq <comp> “freq”] …..
201	Server	Search response	“<filename>” <md5><size><bit rate><frequency><length> <nick><ip>
202	Server	End of search response	
203	Client (to client)	Download request	<nick>”<filename>”
204	Server(to client)	Download ack	<nick><ip><port>” <filename>”><md5> <linespeed>
500	Client	Push file to me (firewall problem)	<nick>”<filename”>
501	Server	Push ack	<nick><ip><port> ”<filename>”<md5> <speed>

Anatomy of a download (A download from B)

- A send a 203 to the server
- A receives a 204 message from client
- A connect (tcp) to ip/port contained in the previous message
- B accept and respond with ASCII char “1”
- A sends the string “GET filename nick offset”
- B answers with the file length or an error message
- (If the download is started) A notifies to the server a 218 message
- (Likewise) B notifies the server a 220 message
- (When the download finished) A notifies the server a 219 message
- (Likewise) B notifies the server a 221 message

Anatomy of a download behind a firewall (A download from B who is behind a firewall)

- A sends a 500 message to the server, witch sends a 501 message to B (with A ip – port)
- B connects to A according to the data in the previous 501 message
- A sends the ASCII char “1|”
- B sends the string “SEND mynick filename size
- A returns the byte offset where to start or an error message
- (If the download started) A notifies the server a 218 message
- (Likewise) B notifies a 220 message
- (When the download finished) A notifies a 219 message
- (Likewise) B notifies a 221 message

3.2 Gnutella

Type	Payload descriptor	Description	Contained Information
Ping	0x00	Announce viability and probe for others peers	None
Pong	0x01	Response to a ping	Ip – port of the responding peer; number and total kb of the shared files
Query	0x80	Search request	Minimum network bandwidth of the responding peer and search criteria
Query hit	0x81	Returned by peers that have the requested file	Ip port abd network bandwidth of the responding peer; number of results and result set
Push	0x40	File request for peers behind a firewall	Peer identifier; index of the requested file; Ip port where to send

Gnutella packets are modelled according to a general pattern showed below

Description header: it appears in front of every message

Descriptor Id	Payload descriptor	TTL	Hops	Payload length

Descriptor id : an unique identifier of the packet
Payload descriptor: the type of the message (es: 0x81)
TTL: number of times the message will be forwarded before being discarded
Hops: number of hops performed so far
Payload length: length of payload area of the message

3.3 Message Format

Ping: Description header + payload with a value of “0x00”
Pong: Description header (payload description 0x01) +

port	IP address	Number of shared files	Number of kb shared

Port: port where the responding peer is able to accept connection
IP address: the responding peer ip’s address
Number of shared file: number of resources shared
Number of kb shared: dimension of the shared files

Query: Payload descriptor +

Minimum speed	Search criteria

Minimum speed: the minimum network bandwidth (in kb/s) of the responding peer
Search criteria: a string indicating the required matching

Query hit descriptor: Payload descriptor +

Number of hits	port	IP	speed	Result set	Peer identifier

Number of hits: number of the matchings found and indicated in the result set
Port: the port where the responding peer is able to accept connection
IP address: the IP address of the responding peer
Speed: bandwidth of the responding peer
Peer identifier: unique identifier of the peer
Result set: number of hits record of this structure

File index	File size	File name

- File index: a number assigned by the responding peer to correlate a query and a response
- File size: size of the file
- File name: the name of the file

Gnutella download are performed by using the http protocol (get operation) at the ip address and port indicated in a query hit message. Response and status code are communicated through 2xx, 4xx status code.

3.4 Gnutella and firewalls

If a peer cannot be contacted directly, the peer receiving the a push message starts the file transfer and send a particular message to the other peer

Peer identifier	File index	IP address	port

Peer identifier: unique identifier of the peer
File index: unique identifier of the file to push
IP address: IP address of the peer, which the file should be pushed
Port: port of the peer, which the file should be pushed

This system does not work if either the peers are behind a firewall

4 General Architecture of a New Model

4.1 Requirements

As seen, none of the most developed communication systems covers both the deterministic search of another user and the security/anonymity needs. This section defines what exactly are the important simultaneous requirements that remain uncovered by the existing models.

The amount of security a user may need ranges from the simple certification of the identity of another user to the complete anonymity about himself and the partner the user is looking for and communicating with. At the maximum degree of privacy, provided that some information are mandatory to be sent if the search of a friend passes through a central server, the use and security needs to be assured are the following:

- Finding friends on the internet to communicate with them. This problem is especially due to dynamical IP addressing; in fact this work is though to be a step toward a good answer to defend the privacy when the search of a friend in the internet is not trivial;

- Making accessible the information that a user is available only to the friends specified from that user; this requirement is a purely privacy problem, that the model has to satisfy;

- Forbidding to anyone to identify the user that is looking for another one; this also is a privacy problem, that preserves the anonymity of a couple of user that are looking for each other;

- Guaranteeing that an available user can be found in the 100% of the searches; this is a strong requirement from an organisational point of view, concerning the server apparatus when the system scales on a large ensemble of users;

- Certifying the identity of a user available for another; this is not a trivial point, and it depends on the specific needs of a user; a certification of a user requires trust in the administrator of the system, but sometimes this is not suitable; in any case the total blind server problem has to be solved for ones requiring this high level of privacy and security.

These requirements obviously suggest the implementation of a central server application to let the research deterministic, but constraint to a special strategy to preserve the anonymity, the certification of the identity and the availability only to allowed friends (invisibility).

4.2 Matching procedure scheme

This section contains the description to the logics of the procedure that allows two users to match and connect. The first choice is to guarantee the security and the certification of a communication by use of two asymmetric keys. In order to build the matching scheme by assuring the requirements let’s check two opposite strategies and find different solutions and security/privacy compromises.

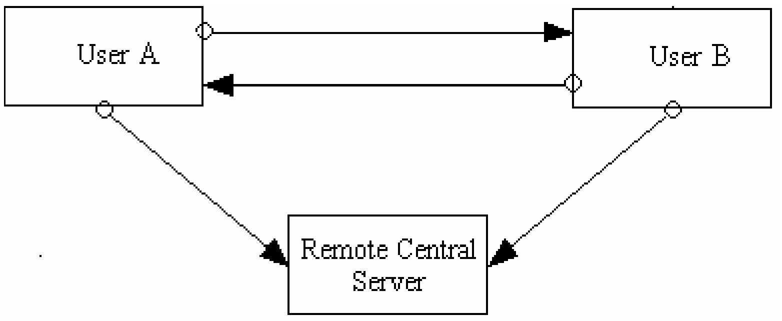


Figure 4.1: The general architecture for the matching of two users.

Suppose the user A wants to communicate with the user B. To match, they:

- Need to recognize each other and
- This must happen under encryption.

Suppose that A connect first. He has to send its identification code (let's call it ID) and its IP address.

ID.IP

Two functions must be applied to this couple of strings to verify the required properties:

1) Only user B must read it, so they have to be signed by the public key of B
2) The user B must be assured that the communication really comes from A, so they have to be signed by the public key of A.

The choice of the order of application of these two functions has to be applied is not trivial and needs at least a short discussion. Three models are discussed in the following.

Model a

Suppose that the string is signed first by the private key of A and then by the public key of B. This can be represented in this way:

$[IP(A)]_{Pr(A)}]_{Pu(B)}$

The ID string disappears because the identity is now provided directly by the signature of A. In this case the priority is that only the user B may read the content of that string, and has to verify that the communication really comes from A. This has the following advantage:

i) Nobody except B knows that A is online and the server is completely blind to the search.

This choice shows also this critic point:

ii) The server system is not able to certify that all the user connected are who they say they are; furthermore it is not able to select which is the information useful for B, so B is devoted to find it.

Model b

Suppose now the opposite situation, that the string is signed first by the public key of B and then by the private key of A. This can be represented in this way:

$$[[IP(A)]_{Pu(B)}]_{Pr(A)}$$

Now the priority is that the user A is a reliable user. This has the following advantage:

iii) No users are somebody else from what they say

The critic point is now that:

iv) The system could be enforced in a nasty way to tell which connection is associated with that reliable user so the privacy is violated

What happens after the connection of the first user A is that the server stores the information in a database, and this information is visible only for the user B. When the user B connects, he tries to open all the strings of the database, to find the user A.

This search now differs significantly in the two models: in the model **a** the server is blind to the content of its database, so it has to send all the strings of the connected users to the user B to let it find friends (for instance the user A) by opening the strings one by one. The amount of traffic can be significantly high if many users are announced in the same server. In the model **b**, the user B remains anonymous but announces to the server that it is looking for A, so the server sends the string certified by A.
This second solution (model **b**) should save a big amount of time and let the transmission between the server and the users very little. Unfortunately the second model is intrinsically weak, especially for an uncovered privacy violation: if a user C is looking for A but A does not want to be found except that from B, the model **b** is not confidential enough, unless each user defines a public/private key pair for each friend.
A little more complicated solution is required, that partially extends the model **b**.

Model c
Suppose that the string transformed by the model **b** is associated to a second string such that:

$$[PIN(AB),[IP(A)]_{Pu(B)}]_{Pr(A)}$$

Where PIN(AB) is an identification code known only by A and B. In this case, when B connects, it sends the request for the connection PIN = xyz and the crypt IP of A is sent by the server.

The differences between the models **b** and **c** are recalled in Table II. From many points of view, the second is preferable. There are still risks due to the fact that the server is not blind (contrarily to model **a**), but this choice represents the only way to avoid an enormous amount of traffic. In any case this ability of the server may be used to preserve the invisibility of the users for the users they do not want to match.
In the latter, the worst thing that may happens, even in a strong nasty way, is the determination by an exploit of the server that the user with public key of A is looking for the user with public key of B.

Model	**A**	**B**		**c**
Server	Blind	Smart	Smart	Smart

Privacy Level	High	Insufficient	Medium	Medium
Number of Private Key Pairs Per User	*n*	1	*n*	1
Covers	Total Anonymity	Large Scalability	Good Scalability	Large Scalability
Limiting factors	Large amount of traffic	Privacy risks, trust in admin	Key pair multi-plication, trust.	Trust in admin

Table II: comparing the main features of the model ***a****,* ***b*** *and* ***c****.*

There is a main difference between the models **a** and the model **c** (**b**). The model **c** requires trust in the administrator of the server, while the model **a** does not. In conclusion, for a private and limited number of users, the model **a** configuration is the only suitable, while for a non-paranoid generic user the **c** is the most versatile. Furthermore, the models **b** and **c** require that the users send their public key too.

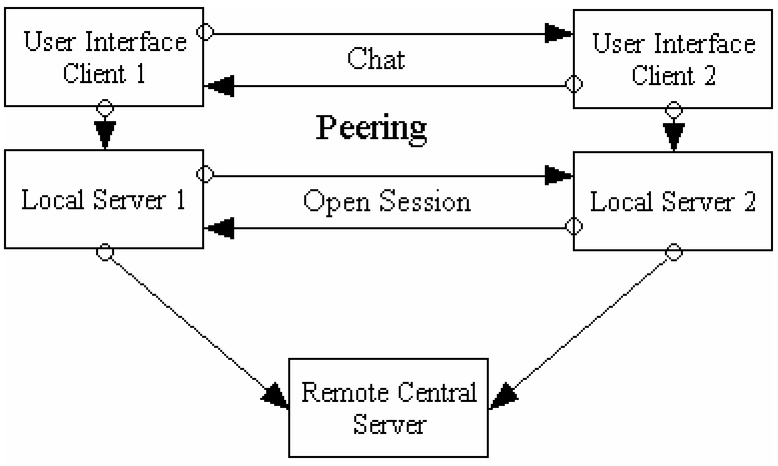


Figure 4.2: The general architecture of the UIC and the LC.

To avoid an exploit based on the brute force reproduction of the string containing the IP together with the known PIN, by trying all the IP and comparing the obtained string with the transmitted one, an additional string can be attached, for instance the port number, or something else [16].

5 The user client application specifications

5.1 Architecture

The client has to be split in two logical layers. They correspond to two different applications. One is the User Interface Client (UIC), one is the Local Server (LS). The logical architecture is the following:

- The P2P socket links the LS of two users to manage the start-up of a session.

- A LS is delegated to the opening of the connection with a remote LS, to crypt/decrypt the messages to certify and recognize the session.

- Each user communicates by the UIC to its own LS. The UIC is the interface to manage the ‘friends’, the permissions and the I/O of the messages.

- Once the session is started, a new window of the client manages separately each conversation

- A management of the certificates is provided: any friend may be contacted by any of the available certificates, or new certificates may be created.

A secure text database is required to maintain the association between the local nicks (LN) given by the user to identify the friends and their public key. The friends are also dynamically associated to an IP, which can also be stored in the database.

5.2 Actions of the UIC and UIC-LS protocol

The UIC provides the following actions:

- Manages the local server by an opportune protocol
- Pops-up windows to communicate with other remote clients after the session has been solved and established by the local server

The protocol the LS supports when the UIC communicates with it is as follows:

1.	[*iptemp* \LN\ \IP\]	Manual association of a LN to an IP
2.	[*ipstatic* \LN\ \IP\]	Store a static IP in the database
3.	[*friend* \LN\ \key\]	Associate an to a public key in the database
4.	[*delete* \LN\]	Delete a LN
5.	[*allow* \LN\]	Allow the connection with LN
6.	[*forbid* \LN\]	Forbid the connection with LN
7.	[*createkeys*]	Create a public and private key
8.	[*setpin*]	In the model **c**, this function is required to match friend through a central remote server

5.3 Actions of the LS and LS-LS protocol

Notice that this section concerns the communication with another LS and or with the eventual central remote server (CRS) to match the research of other users.
The LS provides the following actions:

- May connect to a CRS
- Communicates with another LS
- It establishes the connection once the IP and eventually communication PIN (model **c**) are known.

The features of the LS are:

- Manages the database of LN, public keys and IP number.
- Obeys to the commands of the UIC
- Maintains the connection with the other LS
- Updates data in the CRS regularly during a session

5.4 UIC Requirements and Specifications

The client supports the following functions:

- A main window shows a friend list. The list contains lines with the following information: 1) The Local Nick. 2) If the IP is available at the moment 3) If the connection from that friends are accepted (the user is invisible for him). The name is written in Verdana (or Times New Roman); if the user is available it is **bold**; if the friends is forbidden to connect the color is red.
- A new window is open by clicking on a friend with the left button of the mouse, for the communication.
- Each communication window is separate in two parts: one, above, displays the communication by appending the lines coming from the user and from the friend; one, under, is needed to write the messages. A message is sent by pressing Enter (*send*).
- To change the status (to invisibility and back) for a friend (*allow*, *forbid*) or delete it (*delete*), the right button is used by clicking on a friend.
- The other functions (*iptemp, ipstatic, createkeys, friend*, *setpin*)

5.5 User local database

The database is a text file whose lines contain the information of the users. The database is saved when:

- A static IP is saved (*ipstat*)
- An IP is found by calling a CRS
- A LN is saved with its public key (*friend*)
- A friend is deleted (*delete*)
- A pin is set (*setpin*)

The format of the database is:

```
localnick|publickey|IPaddress
```

The first and the second field are mandatory; is no IP address is know then the default value is 0.0.0.0 .
The database is crypt by a password.

The data of the database are saved in an easy-to-export small size file that can be easily moved from a PC to another, so the user is base independent.

6 The client-to-server communication

As discussed, one has to choose if the priority is the highest possible robustness of the system or the light amount total outgoing traffic from the CRS to the users. This obviously depends also by the number of users one wants to deal with.
The communication between LS and CRS reduces to a two way reciprocal notifications, described as follows. In all the examples suppose that A is the test user that connects and operates.

Model a

The LS first requires to the CRS the full list of announced users. This allows to match the user he is looking for.

- If it is, the LS finds that some messages are there for him, because it recognizes the content of the first square bracket $[IP(B)]_{Pr(B)}]_{Pu(A)}$, so it decrypts the messages by using the public key of B. After that, the IP address is known so it provides to contact its LS directly.
- If nobody is waiting for him, or the user wants to announce its on-line state to somebody, the LS sends a string $[IP(A)]_{Pr(A)}]_{Pu(B)}$ to announce secretly to B that the user is waiting for him; than it wait. It has to send one for each user he is looking for.

Finally, the LS may cancel an announcement, for example when the user disconnects or becomes invisible to some user.

Model c

The sessions run as follows:

- the LS announce to the CRS a string $[PIN(AB),[IP(A)]_{Pu(B)}]_{Pr(A)}$ with its public key separately so the server may certify the identity of the user; one string is needed for each friend;
- the server stores $[PIN(AB),[IP(A)]_{Pu(B)}]_{Pr(A)}$ using the PIN(AB) as key.
- the server sends a list of strings having some the PINs presented by A (PIN(AB) PIN(AC) PIN (AD)..).
- A can connect to users found in that list
- when user B connects it will find the announcement of the PIN(AB) so it can connect to A.

After the IP has been reached, the session starts as described before.

7 Scalability to distributed server network

The main problem that affects central based systems is that when a large number of users, one has to deal with lot traffic. The high amount of traffic is principally a matter of the server, and it is not for the users. A distributed server network may introduce a way to light the total traffic per server.
The proposed architecture provides a hierarchy of two layers of servers. There is a first layer of servers that are public and that provide the main distributed node that guarantee to a user that the connection is not established with a local island but with the main network. These servers provide:

- (Eventually) the ordinary server functions to exchange information with the users
- The service of listing the announced servers that are available, with their own anonymous users list that the requester can contact to search friends. This function is essentially implemented by an extremely simple database application answering the queries by a list of IP addresses.
- The pinging of each announced second layer server, to certify that their connection is still available

Notice that this function of the main servers requires a further function of the UIC, as:

9. [*frontier*] Changes the flag that governs if the LS tries to contact first a main server or connects directly to a layer 2 server specified in list

The second layer servers provide:

- The ordinary server functions to exchange information with the users
- The function to announce themselves to the first layer servers.

The redundancy of the first layer servers is required to avoid single point failure troubles. In this scheme the advantages are due to a better distribution of the resources, in particular the amount of traffic does not change for the user, but from a server point of view is divided by the number of reachable servers.

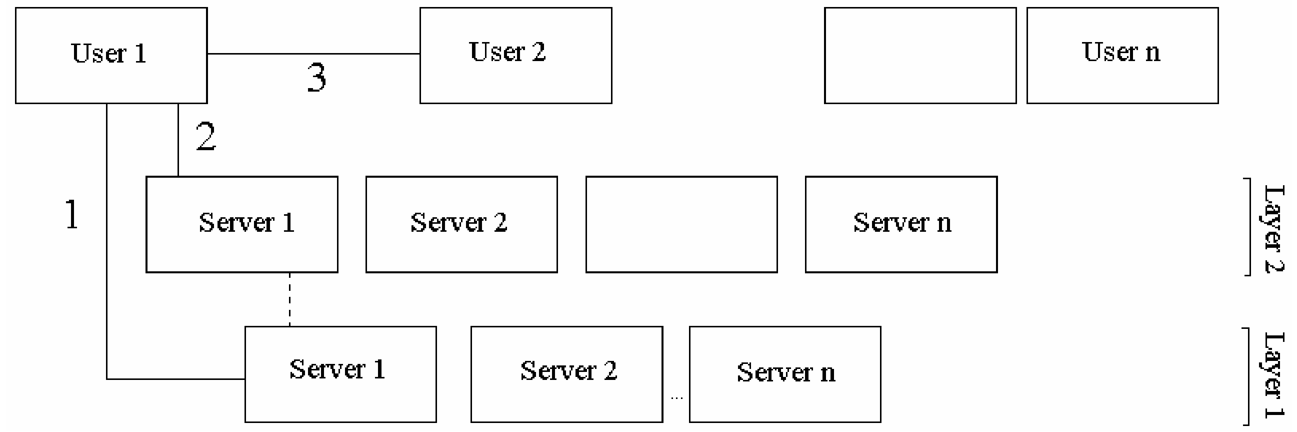


Figure 7.1: The general architecture of the three layers of the client-server hierarchy. The deepest level (Level 1) of servers provides to announces to the users the Layer 2 servers that store information. The user contacts directly them and may find a friend announced on another server.

If *m* users are distributed homogeneously between *n* servers, the number of users announced in each server if *m* / *n*. If announces are *h* bit long, the amount of the traffic incoming into the user segment is:

$$T(user) = n \times (h \times m / n) = h \times m \text{ bit}$$

While the traffic sent to each user from a server is:

$$T(server) = h \times m / n \text{ bit}$$

The conclusion is that this architecture improves the sustainability of the service even in presence of a large amount of users, improving performances especially of the model **a**, that is intrinsically more heavy.

8 References

[1] Jospeph S. Adaptive Routing in Distributed Decentralized Systems: NeuroGrid, Gnutella and Freenet" (English) Proceedings of workshop on Infrastructure for Agents, MAS, and Scalable MAS, at Autonomous Agents 2001, Montreal, Canada

[2] Joseph S. (2001a)
"NeuroGrid and P2P Networks" (Japanese)
Software Design: Network & Systems Magazine

[3] Ozalp Babaoglu, Hein Meling and Alberto Montresor Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems
In Proceedings of the 22th International Conference on Distributed Computing Systems (ICDCS '02), Vienna, Austria, July 2002. Also appears as Technical Report UBLCS-2001-09, University of Bologna, Italy

[4] Alberto Montresor Anthill: a Framework for the Design and Analysis of Peer-to-Peer Systems
In Proceedings of the 4th European Research Seminar on Advances in Distributed Systems, Bertinoro, Italy, May 2001

[5] Karl Aberer, Manfred Hauswirth Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems Tutorial at the 18th International Conference on Data Engineering, February 26-March 1, 2002, San Jose, California.

[6] Karl Aberer, Manfred Hauswirth An Overview on Peer-to-Peer Information Systems
To be pubhlished in Proceedings of Workshop on Distributed Data and Structures (WDAS-2002), Paris, France, 2002

[7] Ian Clarke, Theodore W. Hong, Scott G. Miller, Oskar Sandberg, and Brandon Wiley, "Protecting Free Expression Online with Freenet," IEEE Internet Computing 6(1), 40-49 (2002)

[8] Amr Z Kronfol, FASD: A Fault-tolerant, Adaptive, Scalable, Distributed Search Engine, 2002

[9] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System" in Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, ed. by Hannes Federrath. Springer: New York (2001)

[10] drscholl Napster protcol specification 2000

[11] F.S. Annexstein, K.A. Berman, and M. Jovanovic. Latency Effects on Reachability in Large-scale Peer-to-Peer Networks. In ACM Symposium on Parallel Algorithms and Architectures, Crete Island, Greece, 2001

[12] M. Jovanovic. Modeling Large-scale Peer-to-Peer Networks and a Case Study of Gnutella. Master's thesis, University of Cincinnati, 2001

[13] M. Jovanovic, F.S. Annexstein, and K.A. Berman. Modeling Peer-to-Peer Network Topologies through "Small-World" Models and Power Laws. In TELFOR, Belgrade, Yugoslavia, November, 2001.

[14] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications.* To Appear in *IEEE/ACM Transactions on Networking*

[15] David Liben-Nowell, Hari Balakrishnan, and David Karger, *Analysis of the Evolution of Peer-to-Peer Systems. ACM Conf. on Principles of Distributed Computing (PODC)*, Monterey, CA, July 2002

[16] A special thank to Davide Fais for this remark and for the critical reading of the whole work.