Secure and Mobile P2P File Sharing

Juuso Lehtinen
Networking Laboratory
Helsinki University of Technology
juuso@netlab.tkk.fi

Abstract

In today's Internet, a major portion of the total traffic is generated by peer-to-peer applications such as *Skype* and *BitTorrent*. The amount of this traffic has been increasing since the beginning of this decade. At the same time, mobile use of the Internet has become even more popular. New mobile phones enable people to surf the web and read their mails wherever they are. However, for some reason, mobile peer-to-peer networks have yet to manifest themselves.

This paper studies the special requirements of mobile peer-to-peer (P2P) file sharing. We discuss how the limited resources of mobile devices affect the application design and network architecture choices. We also focus on the security and privacy aspects of mobile P2P networks.

Functionality of P2P networks can be divided into two parts, into content location, and into content download. Security needs of both of these are dealt separately in this paper. Additionally, availability and anonymity requirements of hybrid P2P networks are considered.

Finally, we propose security enhancements for our Session Initiation Protocol (SIP) based mobile P2P architecture that enable secure searches and downloads in mobile P2P environment.

KEYWORDS: Mobile peer-to-peer, peer-to-peer security

1 Introduction

Nowadays, peer-to-peer (P2P) file-sharing is extremely popular in the fixed Internet. The amount of P2P traffic has far exceeded the amount of any other type of application traffic in the Internet. Some studies suggest that 60 – 80 % of all Internet traffic is P2P [10, 1]. Still, P2P file-sharing has not established itself as the killer application in the mobile domain. This has been largely due to limited capabilities of the mobile devices, such as processing power, memory capacity, and network bandwidth. However, as the mobile devices become more capable we can wait to see the P2P applications show up in the mobile domain too.

Traditionally, in the fixed Internet, P2P file-sharing networks have been open for everyone. The content that has been shared in these networks has been accessible to every user in the network. In the public P2P networks, there has been no possibility to restrict access to some files to some specific user group. On the other hand, this has not been a big issue since the main content shared in the P2P networks of the fixed Internet has been music and movies – the kind of content that does not have any privacy requirements from the user's point of view. In certain cases, however, privacy and security features would be handy in P2P networks; for example, if we wanted to share some files inside a closed user group, but not to make these files available for everyone else. Also, currently most of P2P file sharing protocols transmit all their data unencrypted, so that anyone is able to eavesdrop on what is being searched and downloaded. However, some popular P2P protocols, such as *BitTorrent*, are now incorporating options for encrypted file transfers and searches.

This paper studies how secure P2P file sharing can be implemented in a mobile environment. Although the main scope of this paper is on the file sharing P2P networks, many of the techniques presented here can be applied to other P2P uses too, e.g. to P2P instant messaging or video streaming. This paper is structured as follows. In second section, we study special requirements of mobile environment from software perspective. In third section, we evaluate applicability of different P2P architectures for mobile file sharing and review some of the existing mobile P2P implementations. In section four, we will discuss what security features are actually needed in P2P file-sharing and how these features fit into mobile use. In section five, we propose secure enhancements to our SIP based mobile P2P application that, hopefully, will be implemented later this year in the actual application. Finally, in section six, we provide conclusions.

2 Special Issues of Mobile Domain

When designing a P2P application for a mobile handset, we have to take into account the special requirements of the mobile domain. Terminal capabilities are much more limited than in the fixed networks where the terminals have more than enough resources, such as processing power, storage capacity, and network bandwidth. Also, in mobile networks the shared content is most likely user created, whereas in fixed networks the content is mostly professionally created. [12]

Although most of the studies concerning P2P networking have been made with fixed, high capacity networks in mind, research on mobile P2P is slowly emerging. The special requirements and constraints of the mobile platform prevent us from using the P2P protocols developed for the fixed networks in the mobile domain; thus extra work has to be done to enable mobile P2P communication.

2.1 Memory Size

It is important to minimize the memory usage of the application, because modern mobile phones usually have multiple programs running concurrently, and these programs have to share the limited memory. Although the memory capacity is slowly increasing, it is still one of the scarcest resources on the mobile phone.

Memory-use minimization means that we generally have to minimize state information in the application. For example, holding information for hundreds or thousands of peers (like in *Freenet* [5]) is not a good idea in the mobile application. All in all, use of encryption and other security measures should not increase software's memory footprint.

2.2 CPU Performance

As with the memory, the same restrictions go with the CPU cycles, thus usage of complex algorithms should be avoided if possible. Although, thanks to Moore's Law, the CPU speed is a less of an issue with the current mobile phones than it used to be. The CPU usage also directly correlates with the battery usage, as the CPU power consumption depends on the CPU load.

2.3 Access Network Parameters

Another big constraint which has to be accounted for when designing a P2P application is the bandwidth usage of the program. The air interface used to access the mobile network has very limited bandwidth, and this bandwidth has to be shared between multiple users in the cell. This is why it is vital to minimize bandwidth usage of the P2P protocol. Using distributed P2P protocols, that flood requests to many nodes, is inherently a bad idea when considering the bandwidth use. Also, when choosing the signaling and file transfer protocols, we have to minimize the protocol overhead.

2.4 Screen and Keyboard Size

Small screen size of the mobile terminal restricts the user interface design for the application. The P2P application needs to have a lot of information visible to the user, such as the files being downloaded and uploaded, the files in the queue, current bandwidth usage, and the search dialog. Although screen sizes of the mobile phones are increasing and the screens are getting higher resolutions, we are still far away from an ordinary PC screen.

Also small numeric keyboards are not the best ones for inputting extensive textual searches. Some phones have full QWERTY-keyboards, but these are a minority. Thus, UI design has to take the limited features of the input device also into consideration.

2.5 Battery Capacity

Faster processors and larger screens consume the battery faster than before. Current trend of ever decreasing physical size of the mobile terminal is inevitably shrinking the battery too. The battery industry tries to keep up with increased power demands, but best results can be achieved together with the use of modern batteries and low-power CPUs and screens. Also, limiting transmission over the wireless link helps with the energy efficiency.

2.6 Operator Control

Traditionally the network operator has wanted to have control on all the services in its network. Because P2P is inherently distributed, implementing some kind of operator control is a non-trivial issue. However, centralized and hybrid P2P networks make use of super-peers; if the super-peer is under operator control, the operator has virtually full control on the content available in the P2P network. The actual content is still transmitted between the end nodes, but the super-peer acts as a broker for client communications. More control can be added by facilitating special media gateways that inspect the actual content sent between the end nodes.

3 Architecture for MP2P

One way to classify P2P networks is to divide them into structured, unstructured, and centralized architectures. The structured architectures are usually based on distributed hash table (DHT) algorithms. [18]

In the structured architectures, content is found by specifying a unique key for the content, be that hash of the file, hash of the filename, or some other unique identifier. There is no way to perform wildcard searches, and thus structured P2P networks are not generally suitable for file-sharing networks where we want to search content with partial information. Thus, structured P2P networks have had mostly only academic interest and we do not consider their use in file sharing networks any further in this paper.

Unstructured architectures are more suitable for file-sharing networks where we need to have option for wildcard searches. Unstructured architectures can be divided into two sub-architectures: Pure (de-centralized) and hybrid (semi-centralized) P2P architecture. In these architectures users can search for content even though they do not know exactly what they are looking for. Popular file-sharing networks, as well as Skype are based on unstructured P2P architectures.

Third architecture, which is located somewhere between the two earlier architectures, is the centralized architecture. In centralized architecture, there is one centralized directory node, which has global knowledge about the files that the nodes connected to the network are sharing. All searches are done through this centralized node. Figure 1 shows how different P2P architectures relate.

To choose architecture for mobile P2P use, we have to evaluate properties of different P2P architectures.

3.1 Centralized Architecture

The centralized P2P architecture, also often called the Napster architecture, is an architecture where a centralized server, a super-peer (SP), holds information about all the content available in the network. Ordinary peers register with this server and upload information about the files they have to the SP. When a node searches for files, it queries the SP. The SP checks its internal database, and replies the

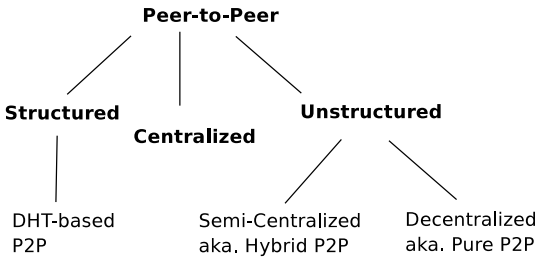

Figure 1: Division of P2P protocols into structured, centralized, and unstructured architectures

query with information about the peers having the requested resource. The querying node then connects directly to a peer that the server just specified. All further communication is directly between these two peers; the server is only vital for searching resources.

The centralized P2P architecture is most suitable for situations where some authority wants to have control on the system. For example, a mobile operator maintaining a P2P service can control the files distributed by the system if it has control on the centralized server. Thus, centralized architecture can be a good choice for mobile P2P.

3.2 Decentralized Architecture

The decentralized architecture, also called the pure P2P architecture, has no centralized node, i.e. no SP. All nodes belonging to the P2P network are equal in this type of architecture. Searches in pure P2P networks are done by forwarding queries from node to node, using a flooded request algorithm (e.g. in *Gnutella 0.4*), which effectively broadcasts the search with limited scope; or by some more intelligent routing method (e.g. *Freenet*), where the query is routed towards the host most likely having the requested file. In both search types, propagation of queries is limited by the TTL field in the queries. If some node has the requested file, it replies to the original querying peer. The reply is usually sent back to the original requester via the same path the request was routed initially. Further communications are directly between the peers or via intermediate nodes, depending whether there is a need for anonymity in the system or not. [16]

Flooded searches create a great amount of messaging overhead in pure P2P networks. Thus, pure P2P networks are really not suitable for mobile networks, unless the network size is really small.

3.3 Hybrid Architecture

The hybrid architecture is combination of the centralized and the pure P2P architectures. The hybrid networks have several super-peers and the end users connect to these SPs as in the centralized architecture. However, the SPs themselves form a decentralized P2P network. The SPs hold information about the resources that the nodes connected to them have. Every SP usually knows only about the nodes connected directly to it, not about the nodes connected to the other SPs. The hybrid architecture is used by many popular P2P applications, such as *Gnutella 0.6* and *Kazaa*. This architecture is also used by *Skype*, a massively popular P2P Internet telephony application, developed by the Kazaa team [2].

In hybrid architecture, queries are forwarded from end nodes to the SP. The SPs query each other if any of their ordinary nodes have the searched resource. If there is a match, peers exchange all further information directly without SP involvement.

3.4 Comparison of Different Architectures

As stated above, different P2P architectures have different properties. The scalability, resiliency, search efficiency and ability to control the network depend on the architectural choice of the network.

For example, the pure P2P architecture does not scale well to a large number of nodes. However, it is the most resilient against node failures, whereas a failure of the centralized server eliminates the whole centralized P2P network. On the other hand, the centralized server may become a capacity bottleneck in the centralized architecture and thus render a large P2P network unusable.

The hybrid architecture is located between the pure and the centralized P2P architectures. It scales quite well and has good resiliency as long as the failed nodes are not functioning as super-peers. The hybrid architecture is also situated somewhere between the other two architectures when considering the search coverage in the network. When a node sends a search out, the request is forwarded to other super-peers which then forward the request onwards. The search coverage is limited by the TTL of the request message, but the coverage is a lot larger in the hybrid architecture than in the pure P2P architecture, thanks to the request only traversing super-peers.

Because both, the hybrid and the centralized P2P architectures have super-peers built into the architecture, some kind of operator control is possible in these networks via the super-peer control.

Comparison between different P2P architectures is presented in table 1.

Table 1: Architectural comparison [12]

Architecture	**Pure**	**Hybrid**	**Centralized**
Network scalability	Low	Very high	Medium
Signaling overhead in super-peer	–	High	Very high
Signaling overhead in ordinary peer	High	Low	Low
Resiliency	Very high	Medium	Low
Operator control	Low	High	Very high
Search coverage	Medium	High	Very high

It can be argued that the centralized and the hybrid P2P architectures are most suitable for mobile P2P networking. If the super-peer is located in the fixed network, the mobile nodes do not have to handle huge signaling loads during searches. All the mobile terminal has to do, is to send

the search to the super-peer it is connected to. The super-peer consults its own database (and other super-peers in case of the hybrid architecture) and sends the reply to the mobile terminal.

The centralized and the hybrid architectures can also provide the mobile operator some control over the shared files. As long as the operator controls the SP, it can enforce sharing policies and ban unwanted files. Of course, some users consider this as a bad feature. Operator control also inherently kills the possibility for any kind of anonymity in the network.

The choice between the hybrid and the centralized architecture is dictated by the overall network design. If the idea is that the network operator controls the super-peer, and that the users of different operators are able to share files, the hybrid architecture is the most suitable. In this architecture every operator can run its own super-peer that connects to the other super-peers ran by the other network operators.

From the viewpoint of the mobile application the centralized and the hybrid architecture look the same. The mobile terminal is still connected to one super-peer; it does not care if the answer is combined from the results of multiple super-peers connected in P2P fashion, or from the database of a single super-peer. Because of this, the same application can be used for connecting to centralized and hybrid P2P networks.

3.5 Current MP2P Systems

Mobile P2P applications are still rare. However, some applications have been implemented as academic efforts, and some of them are available even for public. None of them implements any security measures, such as peer authentication, signaling or content encryption.

3.5.1 Mobile eDonkey

Oberender et al. [17] have developed a mobile P2P file transfer architecture based on the *eDonkey* protocol. Some modifications to the original architecture have been done. This architecture includes modifications to the Index Server, a Crawling Peer and a Cache Peer. The index server keeps track of the popularity of the files in the network, and exports this popularity data to the cache peer. The cache peer stores these popular files, and the crawling peer supports the index server by linking it to other index servers in the Internet. The resulting architecture is something between centralized and hybrid P2P.

The benefit of the cache peer is that the popular files can now be stored in the core network, and they do not have to be transferred multiple times over the air interface. When a file is traditionally transferred from one mobile to another, the data goes over the air interface twice. Reducing this traffic to half is the main benefit of the cache peer.

3.5.2 Symella

New software called *Symella* was released in late 2005. Symella is a *Gnutella 0.6* client that works on *Symbian* smartphones [11]. To our knowledge, it is the first publicly available mobile P2P application. The software enables mobile user to search and download content in *Gnutella* network, but it does not enable users to share anything.

The application does not provide any extra security features over the traditional Gnutella protocol. The searches are flooded in the network and are visible to intermediate peers, and the file transfers are unencrypted.

3.5.3 Mobile Gnutella

Another paper [21] argues that the usual P2P file sharing networks, such as *Gnutella*, are not suitable for the mobile environment due to their bandwidth consuming broadcast nature. Instead, a modified architecture for Gnutella network is proposed where a mobile agent in the fixed network works on behalf of the mobile device. The mobile agent is part of the Gnutella network, where it acts as a normal Gnutella peer, and has vital information like the file-list of the mobile device. The mobile device and the agent communicate using a light-weight protocol. Thus, the mobile agent can handle most of the signaling traffic, such as searches, and direct only download requests to the mobile device. The mobile device can perform the actual file transfer directly with the other end node or, alternatively, the mobile agent can do this on behalf of the mobile device.

3.5.4 SIP based MP2P

We have created a mobile P2P software for Nokia's *Series 60* smartphones. The software uses Session Initiation Protocol (SIP) as its underlying signaling protocol so the application can be used in any SIP aware network. The main emphasis in the application design has been compatibility with future IMS networks. The software is based on hybrid P2P architecture where the super-peer is implemented as an application server, and where multiple super-peers can network in pure P2P fashion. Usage of SIP enables signaling to be routed using SIP Uniform Resource Identifiers (URIs) as node identifiers. Use of SIP identifiers enables seamless mobility because changes in node IP-addresses are abstracted away using the underlying SIP infrastructure. [15, 3].

Our architecture uses two standard SIP methods to implement all of its functionality: INVITE and MESSAGE. Nodes upload information about the files they are sharing to the super-peer in MESSAGE requests. Search requests are conveyed in bodies of INVITE messages that are sent from mobile nodes to the super-peers. The SPs send search replies in the following 200 OK messages. The actual content download is performed by sending INVITE request to the peer that has the file of interest. The Session Description Protocol (SDP) in the message body specifies the hash of the file we want to download. Both, file list updates and search requests encode their bodies in Extensible Markup Language (XML); thus, they can be extended easily in the future.

Currently the software does not implement any security features. One purpose of this paper is to propose security measures for this application that can be implemented later this year.

4 Security in Hybrid MP2P

According to [20], the security problems of P2P systems include: authentication, encryption, privacy and confidentiality, and ability to deal with malicious nodes. A bit different categorization is used by Daswani et al. [7], who organize security issues of P2P data-sharing into four areas: availability, file authenticity, anonymity, and access control.

In file P2P sharing environment we can divide security issues by functionality into two main categories – into security issues of search and into security issues of content transfer. Whereas content searches are done using a fairly static super-peer, the actual content is transferred from peer to peer, where the other peer can be any random, never-seen-before node. We are going to address the search and download security issues, as well availability and anonymity issues as separate sections. We confine our discussion to mobile node to super-peer and mobile node to mobile node communications.

4.1 Search Security

When searching for content the mobile node has to send the search parameters to the super-peer. These parameters might include file name, file type, file size, or some other relevant parameters. Bellovin [4] describes how both Napster and Gnutella may leak information that users consider private such as search strings. This is one of the major problems with most of modern P2P systems – the search messages are sent unencrypted from node to node. This enables simple eavesdropping attacks to reveal what the mobile nodes are searching.

Search requests should be encrypted when they are sent to the super-peer so that eavesdropping the connection between the mobile node and super-peer does not expose search information. However, because super-peer inherently functions as the information database, all searches have to be exposed to the super-peer; thus, the users of centralized and hybrid P2P systems have to trust the super-peer.

Authentication and access control are vital components to enable sharing inside private groups. We can assume that users want to share some content only to their friends or families. This requires implementation of group management mechanisms in the P2P system. Mobile nodes have to authenticate to the super-peer before they can commit searches, so that the super-peer knows which groups it can consult when searching for content. The authentication token can be a simple password, or it can be some more complex structure, such as user certificate. Because we can assume super-peers being fairly static, i.e. the same mobile node has always the same SP, we can use any authentication mechanism used in client/server world between the mobile node and the super-peer. The group management issues are probably worth a paper of their own, so we are not going to study their implementation further here.

4.2 File Transfer Security

File transfers are done directly from peer to peer after we have done the search with the super-peer. Because we assume it is impossible to know in advance with whom we are going to communicate in the future, it is not feasible to have any kind of shared secret established with all the other peers in the network. Thus, authentication between peers creates a problem: how can we authenticate a peer we have never seen?

Nonetheless, authentication between peers is important if we are acquiring some private content, shared only for members of some specific group. The authentication might be done using a group-wide shared secret. However, there is risk that this secret leaks, and compromises whole group security. It is also possible that each member of group knows every else member of group and has established shared secrets with them. However, because of scalability issues, this is probably not the case. One possibility might be using some kind of public key infrastructure where node certificates would be signed by the super-peer, and we could check with the super-peer before admitting the download request if the requesting node is actually member of the group.

It seems that secure and scalable authentication between foreign peers requires some centralized node to mediate the process. That node can be the super-peer or some other centralized server (e.g. *Skype* uses a centralized authentication server [2]).

Another essential feature of file transfer security is encryption. We have to use encryption protocol to secure all traffic between the peers; both, signaling and the actual file transfer. Any standard encryption protocol can be used to encrypt this transmission.

4.3 Availability

To guarantee maximum node availability in the network, the network has to be able to handle Denial of Service (DoS) attacks. Some research has been done considering DoS attacks in decentralized P2P networks; however, this research seems to be concentrating mostly on query-flooding DoS attacks [6].

It is impossible to protect against network level DoS attacks, where node's network capacity is overflooded. However, it is important to protect against application level DoS attacks, where small to moderate number of packets can result in large destruction. One type of DoS attack is amplification attack where malicious node uses other nodes in the network to cause significantly more damage than it could affect with its own available resources [7].

In hybrid P2P networks it might be possible to create amplification attacks for example by exploiting search mechanisms in the network. It is possible that a node spoofs its own address and sends multiple search requests using someone else's address. These requests can result in large replies, and these replies are then sent to the node whose address the attacking node was spoofing. In mobile networks, where the node bandwidth is scarce, these attacks can easily render the target node unusable. To prevent these attacks, super-peers have to implement methods that rate-limit maximum number of searches by one node in certain time period. If search rate of a node exceed this rate-limit, the super-peer should ignore further queries, and thus prevent possible amplification attacks. It is also possible that these attacks are targeted at the super-peer itself to exhaust its resources.

Amplification attacks described above were performed by exploiting the super-peers. However, the application architecture should also consider attacks against availability that consist of only peer-to-peer communications. These attacks can include false download requests that result in node reserving resources for the file transfer in vain, and potentially preventing legitimate download requests.

One general way to prevent attacks against availability, such as those described above, is to make the attacking node perform more work than the node it is attacking. This can be accomplished for example by forcing the node to solve a cryptographic puzzle before it can perform the search or download request [13].

4.4 Anonymity

Hybrid P2P networks are not generally able to provide any kind of anonymity because nodes are using same static super-peers to search for files. And because there are no intermediate nodes between the peers during the downloads, the file transfers partners can be easily identified.

Usually P2P networks that provide good anonymity lack in search features, and generally demand more bandwidth because all messaging is routed through multiple nodes. Thus, creating an anonymous and efficient mobile P2P network might not be feasible.

However, nodes that are concerned about their anonymity can use external anonymizing services, like onion routing services (e.g. *Tor* [9]), when using P2P networks.

5 Secure SIP Based MP2P File Sharing

We propose security features for our SIP based file-sharing application which enable secure searches and file transfers between peers. The basic application is presented earlier in this paper.

A good P2P security architecture is used in JXTA framework. This architecture can be used as a model when implementing secure file sharing over our SIP based application. JXTA is an open P2P platform which can use Transport Layer Security [8] (TLS) for end-to-end encryption. Authentication in JXTA framework is based on X509v3 certificates [22].

5.1 Signaling Security

Because we are using SIP as the signaling protocol we can use any means of the SIP protocol to provide signaling authenticity, integrity, and confidentiality. Securing signaling is done in two steps; first securing hop-to-hop signaling with TLS, and then end-to-end signaling with Secure Multipurpose Internet Mail Extension (S/MIME) [19].

Instead of using SIP URIs to contact the super-peer and other peers we can use SIPS URIs to provide secure, encrypted transport of signaling. Basically this means that all signaling is transmitted using TLS encryption. However, signaling is encrypted not end-to-end, but from hop-to-hop, so that every SIP proxy along the signaling path is able to deduce what is signaled. This is because the SIP proxies need to view certain header fields to route SIP messages correctly. To secure signaling end-to-end we should use S/MIME encryption in the SIP message bodies. S/MIME provides end-to-end confidentiality and integrity for message bodies, i.e. for search strings, download initializations, and for file list updates. Neither the use of TLS or S/MIME should increase the protocol overhead more than some percents.

Authentication of mobile node to the super-peer should be based on a pre-shared secret. The pre-shared secret can be derived for example from mobile terminal's International Mobile Subscriber Identity (IMSI) code which is saved on Subscriber Identity Module (SIM) card and only known by the handset and the network operator. The IMSI code is unique for every subscriber, and it is stored both on the SIM card and in operator's Authentication Centre (AuC) database. Because we assume the network operator is operating the super-peer there should be no security issues using this approach. Also, requiring authentication for all searches makes DoS attacks more difficult because unauthenticated search requests can be ignored. Without mandatory authentication super-peer can be facilitated in amplification attacks against other nodes by including SIP URI of the target node in SIP Contact header.

5.2 File Transfer Security

The actual download process is separate from the signaling (remember we still use secure SIP signaling to start the download) so we can use any industry standard protocol for securing it. Because TLS is already used with SIP signal encryption, it is good idea to use it also for content encryption. TLS provides robust encryption, and using the same protocol in several places reduces the software complexity.

Authentication of remote peers is still under evaluation. Because our architecture uses super-peers for searches, it seems that they are good points for mediating peer authentication. Use of shared secrets for group management is not seen as a good idea because then one bad node can compromise the whole group security.

A probable solution lies in the use of certificates that are signed by the super-peer. The nodes could check each other's authenticity via the super-peer when ever they establish a file-transfer session. Also some kind of Kerberos style authentication might be considered. [14]

6 Conclusion

This paper presented an overview on mobile and secure P2P file sharing. We discussed special constraints of mobile environment such as limited CPU power, memory capacity, and network bandwidth. We concluded that hybrid P2P network is the best architecture for mobile use due to its low signaling overhead and ability to give operator some control on the service.

We also saw that we need to secure both content searches and the actual downloads in P2P networks. We saw that if searches are done using static super-peer, the mobile nodes can have some shared secret with the SP in advance and thus

securing searches is not that much of an issue. However, because downloads are done from random peers, it is much more difficult to secure this connection. Authentication between peers has to be based on Public Key Infrastructure (PKI), or some centralized authentication server.

Finally, we proposed modifications to our SIP based P2P architecture which would enhance the application security. We proposed using TLS, and S/MIME encryption for all the signaling, and TLS for the actual file downloads too.

Providing security in a mobile P2P environment is a complex issue. Important future research subjects are how to implement group management and how to secure communication between peers in a fully distributed environment.

References

[1] N. B. Azzouna and F. Guillemin. Experimental Analysis of the Impact of Peer-to-Peer Applications on Traffic in Commercial IP Networks. *European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services*, November – December 2004.

[2] S. A. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. September 2004.

[3] N. Beijar, M. Matuszewski, J. Lehtinen, and T. Hyyryläinen. Mobile Peer-to-Peer Content Sharing Services in IMS. In *The International Conference on Telecommunication Systems, Modeling and Analysis 2005, ICTSM2005*, Dallas, Texas, USA, November 2005.

[4] S. M. Bellovin. Security Aspects of Napster and Gnutella. 2001 Usenix Annual Technical Conference, Invited talk, June 2001.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. White paper, 1999.

[6] N. Daswani and H. Garcia-Molina. Query-Flood DoS Attacks in Gnutella. In *ACM Conference on Computer and Communications Security*, 2002.

[7] N. Daswani, H. Garcia-Molina, and B. Yang. Open Problems in Data-Sharing Peer-to-Peer Systems. In *ICDT 2003*, January 2003.

[8] T. Dierks and C. Allen. The TLS Protocol. RFC 2246, January 1999.

[9] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium*, August 2004.

[10] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos. Is P2P Dying or just Hiding? In *Globecom 2004*, November – December 2004.

[11] I. Kelényi and B. Molnár. Symella. *http://symella.aut.bme.hu/*. Referenced: March 21st 2006.

[12] J. Lehtinen. Design and Implementation of Mobile Peer-to-Peer Application. Master's thesis, Helsinki University of Technology, 2006.

[13] M. Ma. Mitigating Denial of Service Attacks with Password Puzzles. In *ITCC 05*, 2005.

[14] Massachusetts Institute of Technology. Kerberos: The Network Authentication Protocol. *http://web.mit.edu/kerberos/*.

[15] M. Matuszewski, N. Beijar, J. Lehtinen, and T. Hyyryläinen. Mobile Peer-to-Peer Content Sharing Application. In *IEEE Consumer Communications and Networking Conference, CCNC2006*, Las Vegas, Nevada, USA, January 2006.

[16] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical report, HP Labs, Rutgers University and University of California at Santa Barbara, 2002.

[17] J. O. Oberender, F.-U. Andersen, H. de Meer, I. Dedinski, T. HoSSfeld, C. Kappler, A. Mäder, and K. Tutschku. Enabling Mobile Peer-to-Peer Networking. In *Lecture Notes in Computer Science*, volume 3427, pages 219 – 234, 2005.

[18] S. Rieche, K. Wehrle, O. Landsiedel, S. Götz, and L. Petrak. Reliability of Data in Structured Peer-to-Peer Systems. In *International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 108 – 113. IEEE, 2004.

[19] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, June 2002.

[20] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using SIP. In *NOSSDAV '05: Proceedings of the international workshop on Network and operating systems support for digital audio and video*, pages 63–68, New York, NY, USA, 2005. ACM Press.

[21] T. H. ting Hu, B. Thai, and A. Seneviratne. Supporting Mobile Devices in Gnutella File Sharing Network with Mobile Agents. In *Proceedings of the Eight IEEE International Symposium on Computers and Communications (ISCC'03)*. IEEE Computer Society, 2003.

[22] W. Yeager and J. Williams. Secure Peer-to-Peer Networking: The JXTA Example. *IT Professional*, 4(2):53 – 57, March – April 2002.