A Comparison of Mobile Peer-to-peer File-sharing Clients

Imre Kelényi[1], Péter Ekler[1] , Bertalan Forstner[2]
PHD Students[1], Assistant Professor[2]
Budapest University of Technology and Economics
Department of Automation and Applied Informatics

1. ABSTRACT

File-sharing on mobile phones is becoming widespread. With the recent introduction of the first mobile BitTorrent clients, any mobile user can access content shared via BitTorrent. We created two BitTorrent clients, one for Symbian OS, and the other for Java-enabled mobile devices. In this paper we overview their key characteristics, compare their differences and discuss the issues we faced during the implementation of the applications. We also present the results of our performance measurements with the clients.

2. INTRODUCTION

BitTorrent [1] is a Peer-to-peer file-sharing technology that is widely used on fixed computers. In contrast with centralized file-transfer solutions, such as FTP, it enables the users to share files in a cooperative way. With BitTorrent, while several peers are downloading the same file at the same time, they also upload pieces of that file to each other. This redistributes the cost of upload to downloads.

Mobile phones with advanced connectivity features and considerable processing power are becoming widespread. Furthermore, due to the increased multimedia capabilities and larger storage capacity of these devices, the amount of information handled and generated by applications running on mobile phones increases every year. There is significant need for effective file-sharing solutions. Since BitTorrent requires multi-threaded client software that performs complex operations, only recent mobile devices have enabled deploying BitTorrent clients into mobile environments [2].

We released SymTorrent, the first BitTorrent client for mobile phones, in 2006. It is a generic BitTorrent client that allows the users to both download and share files on their mobile phones. It was written for Symbian-based smartphones in native C++.

Following SymTorrent, which was mainly used on smarthphones, we decided to bring the technology to casual mobile phones as well. In 2007, the first version of our Java-based BitTorrent client was released. It is referred to as MobTorrent and can be used on any phone that supports J2ME with CLDC 1.0 (Connected Limited Device Configuration) and MIDP 2.0 (Mobile Information Device Profile) [3]. It also requires JSR 75 that allows J2ME applications to use the file system. In this paper we summarize our experiences with these two mobile BitTorrent clients. We examine their key characteristics, including both their advantages and disadvantages. We show how the Java version performs compared with the native implementation. Furthermore, we also discuss the most significant issues we faced during the design and implementations of the applications.

The rest of the paper is organized as follows. In Section 3, we briefly overview the related research done on BitTorrent, emphasizing its mobile-specific aspects. In Section 4, we introduce the two related BitTorrent applications and discuss their key characteristics. Section 5 focuses on the comparison of the two implementation, including the results and interpretation of our measurements. Finally, Section 6 concludes our work.

3. RELATED WORK

In the last years, BitTorrent has emerged as a very scalable peer-to-peer file distribution mechanism. While early measurements and analytical studies verified the performance of BitTorrent, they also raised questions about various metrics (upload utilization, fairness, etc.).

Bharambe et al [4] presented a simulation based study of BitTorrent. Their goal was to deconstruct the system and evaluate the impact of its core mechanisms. Their evaluation mainly focuses on peer link utilization, file download time and fairness amongst peers in terms of volume of content served. Their results confirm that BitTorrent performs near-optimally in terms of uplink bandwidth utilization, and download time except under certain extreme conditions. They also showed that low bandwidth peers can download more than they upload to the network when high bandwidth peers are present.

Pouwelse et al [5] presented a detailed measurement study over a period of eight months of BitTorrent. They presented measurement results of the popularity and the availability of BitTorrent, of its download performance, of the content lifetime, and of the structure of the community responsible for verifying uploaded content. The results showed that the system is quite popular, but the number of active users in the system is strongly influenced by the availability of the central components. They also found that 90% of the peers experienced speeds below 65 kB/sec. From the lifetime point of view, they showed that only 9,219 out of 53,883 peers (17 %) have an uptime longer than one hour after they finished downloading. For 10 hours, this number has decreased to only 1,649 peers (3.1 %), and for 100 hours to a mere 183 peers (0.34 %).

Little research has been done on the mobile implementation of BitTorrent. [2] introduces an architecture for using BitTorrent in a completely mobile environment. [6] concludes our experiences with the Java version of the client.

Nurminen et al. [7] used SymTorrent and analyzed practical content sharing scenarios on mobile devices. They showed that mobile P2P file-sharing is feasible from the power consumption point of view and with better implementation and protocol extensions, mobile applications can be made more energy-efficient.

4. BITTORRENT ON MOBILE DEVICES

Bringing the BitTorrent technology to mobile devices is a challenging task due to the limited resources available on mobile phones. The situation is more difficult if the target devices are not only smart phones but also mainstream phones with even less resources.

When we speak about P2P solutions on mobile devices it is important to emphasize that in order to become a full member of a P2P community, the mobile device must fulfill the following specifications:

- The mobile device has to be able to connect to the network via the specific P2P protocol.
- It has to be able to download and upload content.
- Publishing feature, which means that mobile users should also be able to create and share new contents to the P2P community.

If the device is only able to connect to the network and download content but cannot upload, then it is not a full member of the community. This behavior is not highly appreciated by the other members. In many cases (depending on the specific P2P implementation) the P2P network and the algorithms detect this selfish behavior and punish the specific peers with lower service rate or worse service quality.
However, if a mobile phone is not able to become a full member of the community because of its inherent limitations, then a good solution would be to offer extensions like server support, which would enable mobile devices to become valuable members of the P2P network.

4.1. *SymTorrent*

SymTorrent is the first and, at the point of writing, the only BitTorrent client for Symbian OS. Our main goal was to transfer the BitTorrent technology to a mobile platform and demonstrate the possible use cases of BitTorrent-based file sharing on a real device. In addition, we developed some new concepts during the development which resulted in an integrated client-tracker application. SymTorrent not only works as a standard BitTorrent client, it also has its own built-in tracker. Running a tracker on a mobile phone may seem a bit bizarre at first but it can have several interesting use cases. Sharing files instantly between a small group of users without depending on external servers is just one example.
Since SymTorrent was written in native C++, we did no have difficulties with accessing the more advanced services of the platform. Symbian OS is a multithreaded operating system that is capable of hosting applications using several sockets, file-access and complex user interface.
Since Symbian-based phones use different screen sizes and input methods, we implemented the UI-independent parts of the application in a separate DLL. This way of porting to different devices is much easier.
During the last year, SymTorrent has been downloaded more than twenty thousands times. Most users employ it as standard BitTorrent client for downloading files with their mobile phones through GPRS or WLAN.

4.2. *MobTorrent*

MobTorrent is a complete J2ME-based BitTorrent implementation supporting both downloading and uploading. Most of today’s mobile devices support J2ME

applications. There are already several popular applications for the platform. Furthermore, it is very reliable: the signing procedure of the applications protects the users against unsafe or malicious software.
Although software development is easier in J2ME than in Symbian C++, we still faced several difficulties during the implementation of MobTorrent. These issues are related to the J2ME implementation of the different mobile platforms.
The first issue is related to the file handling on J2ME, which differs from the general Java file handling. The source of the problem is that, on J2ME, we can access the files only via input and output streams which are slower than other file handling implementations.
In BitTorrent, we download the content in separate pieces. A general piece size is 64 KB. Pieces are downloaded in separate blocks; in this case the block size is 16 KB. In SymTorrent, after the whole piece is downloaded the application reads it from the file system and does the hash calculation for it. This solution was very slow in MobTorrent. Investigating this issue, we measured the speed of reading a 64KB size piece from different position of a file and calculating its hash value.

Table 1
Piece read and hash calculation performance

	Begin		**Middle**		**End**	
millisecond	Read	Hash	Read	Hash	Read	Hash
file1 2 MB	1097	978	4398	978	7103	978
file2 7 MB	1109	978	10847	978	20210	978
file3 10 MB	1150	978	14478	978	27214	978

Table 1 illustrates how much time it takes to read the piece from a file and to calculate the hash value, the values are in milliseconds. The measurements were made on a Nokia N93 device but the rates were the same on other devices, therefore, we omit these results here. We chose file sizes which are typically used on mobile devices for mp3 or other multimedia files. We also examined reading the piece from different positions of the file. In Table 1, we can see that the real bottleneck is not the hash calculation, but the reading of the piece of data from the file system. The reason for that is the file handling of J2ME. We can read from a file only via an InputStream, which does not enable direct seeking. The only way to seek in the file is to read it until the right position is reached.
In order to avoid this performance penalty, we calculate the hash value incrementally when a block of a piece arrives; thus, we do not have to read it from the file system after the whole piece arrived. To achieve this, we must query the blocks of a piece in the right order, which is feasible because if a peer reports us that it has the whole piece, then we can query them in the right order.
The other problem is the network handling on J2ME. The BitTorrent protocol [1] uses HTTP for the communication with the tracker and TCP connections for transferring data between the peers. We realized that there are limitations about how many connections can be kept open at the same time. On S40 devices, this number was 9; while on S60 devices we did not find any limitations. Downloading via 9 parallel connections is adequate in many cases (if the number of peers that provide certain content is limited or if the peers are fast and just few of them are able to

support fair download rate). However, this can be a bottleneck when a popular content is available from many peers and that particular peers are slow. Section 5 introduces measurements about the performance of SymTorrent and MobTorrent. We compare how the previously mentioned issues decrease the performance of MobTorrent if the bandwidth is high enough.

5. MEASUREMENTS AND RESULTS

We tested SymTorrent and MobTorrent in a real environment. Our test file was the torrent of the original BitTorrent client: bittorrent441.torrent.

Table 2

Download speed (kB/sec) comparison in real environment

	3G	WLAN
SymTorrent N91	50	150
MobTorrent N91	48	79

In a real environment the download speeds depends on several factors that are beyond our control. We chose a popular torrent and ensured that all of the peers were alive during the download in order to avoid delays of the long socket timeout. In Table 2, we can see that with 3G network connection, the performance of the J2ME applications was comparable with the Symbian.
With WLAN connection, SymTorrent was much faster. It is due to the reasons we mentioned beforehand and because MobTorrent usually manages to connect to only 5-9 peers, while SymTorrent can download from 9-15 peers or more.
Besides the quantitative performance comparisons there are a number of other dimensions to compare MobTorrent and SymTorrent. From the architecture point of view, there are only a few differences between the applications. The main one is that MobTorrent uses the standard thread implementation of J2ME while SymTorrent uses the Symbian-specific active objects framework for running and scheduling parallel tasks. Symbian also allows detailed control over the user interface while in J2ME the UI has to be compiled with a more limited set of widgets. These are examples for the conceptual differences of J2ME and Symbian programming. With Symbian programming, we can reach low level APIs of the operating system but have to work in a non-standard environment. With J2ME we have less control over the details and depend on the availability of JSR for key activities. However, in return, working with J2ME is easier due to the standard programming approach. Furthermore, most of the J2ME development frameworks have tools (e.g. for UI design) that make the application development faster. Figure 1 shows screenshots of the UI of both SymTorrent and MobTorrent.
From the software size point of view the source code of MobTorrent is 21500 lines of code, and SymTorrent has approximately 31000 lines. The development time of MobTorrent was around two months for one person and for SymTorrent it was around four months. We can say that it was easier to develop the MobTorrent, but

the real difference should be a bit lower because experiences gained in SymTorrent development helped a lot in MobTorrent development.

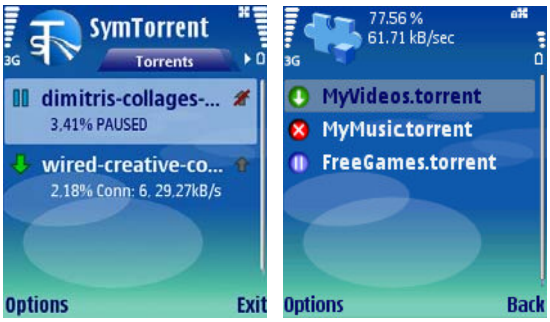

Figure 1
The main view of SymTorrent and MobTorrent

6. CONCLUSIONS

BitTorrent is no more a privilege of desktop computer users. With the introduction of SymTorrent and MobTorrent, mobile users have also started to harness the benefits of distributed file-transfer. In this paper, we gave a brief overview of the currently available mobile BitTorrent technologies and showed how they perform compared with each other. As expected, the native version performs better in most of the cases; however, it is available for Symbian-based smartphones only. MobTorrent has a few shortcomings, including some compatibility issues and the socket-related problem, but most of these are related to Java and not our implementation. We expect future versions of the Java virtual machine to fix most of the listed issues.

REFERENCES

[1] BitTorrent specification, Jan. 03, 2008. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[2] Imre Kelényi, Bertalan Forstner: Deploying BitTorrent Into Mobile Environments. WSEAS European Computing Conference 2007 (ECC'07), Athens

[3] Java Micro Edition technology description, Jan. 03, 2008. [Online]. Available: http://java.sun.com/javame/technology/index.jsp

[4] A. R. Bharambe, C. Herley, V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Network's Performance Mechanisms", INFOCOM'06, 2006, Barcelona, Spain

[5] J. Pouwelse, P. Garbacki, D. Epema, H. Sips, "The BitTorrent p2p file-sharing system: Measurements and analysis", IPTPS'05. 4th International Workshop on Peer-To-Peer Systems 2006, Ithaca, New York, USA

[6] P. Ekler, J. K. Nurminen, A. J. Kiss "Experiences of implementing BitTorrent on Java ME platform", CCNC'08. 1st IEEE International Peer-to-Peer for Handheld Devices Workshop 2008, Las Vegas, USA

[7] J. K. Nurminen and J. Nöyränen, "Energy-Consumption in Mobile Peer-to-Peer – Quantitative Results from File Sharing," in Proc. of Fifth IEEE Consumer Communications & Networking Conference (CCNC) 2008.