Dessy: search and synchronization on the move

Eemil Lagerspetz, Sasu Tarkoma
Helsinki Institute for Information Technology HIIT
University of Helsinki
P.O. Box 68 FI-00014 University of Helsinki, Finland
Email: firstname.lastname@hiit.fi

Tancred Lindholm
Helsinki Institute for Information Technology HIIT
Aalto University
P.O. Box 5400 FI-02150 Espoo, Finland
Email: tancred.lindholm@hiit.fi

***Abstract*—Current smartphones have a storage capacity of several gigabytes. More and more information is stored on mobile devices. To meet the challenge of information organization, we turn to desktop search. Users often possess multiple devices, and synchronize (subsets of) information between them. This makes file synchronization more important. This paper presents Dessy, a desktop search and synchronization framework for mobile devices. Dessy supports synchronization of search results, individual files, and directory trees. It allows finding and synchronizing files that reside on remote computers, or the Internet. The contributions of this paper include an energy usage evaluation of the system. Dessy is closely integrated with the Syxaw file synchronizer, which provides efficient file and metadata synchronization, optimizing network usage.**

I. Introduction and Related Work

The storage capacity of smartphones is growing, while file search functionality on mobile platforms remains simple. Existing desktop search software allows searching the content of files on desktop computers, but not mobile platforms. Some of the most known examples of desktop search are Apple's Spotlight[1], the Tracker project[2] and Copernic Desktop Search[3]. Constraints such as limited battery life, CPU power and memory pose additional challenges to the design of software on mobile platforms. Crawling and indexing the file system on the mobile may not be practical.

This paper introduces the design of a desktop search and synchronization system for mobile devices, called Dessy. We evaluate the performance and energy usage of our prototype system. Dessy finds files by their content, metadata, and context information. It enables searching for files on the local file system, remote computers, and the Internet. Dessy supports remote searching and index synchronization. This offloads the indexing and crawling task from the mobile device to a more capable machine.

Dessy differs from most desktop search software by operating on multiple mobile platforms. Dessy can be used on desktop computers, MIDP 2.0[4] / CLDC 1.1[5] smartphones, and Java Foundation Profile PDAs.

Finding files has also been approached using virtual directories [1]. The use of context information and user–friendly metadata was explored by Cuttrell et al [2]. In Dessy, queries are represented by virtual directory paths.

To allow access to the user's data on the desktop machine, file synchronization can be used. Many synchronization systems for mobile devices have been presented [3], [4]. The SyncML synchronization protocol specification [5] accommodates regular file synchronization, but it has been used mostly for personal information management. For synchronization of files, a well–known system is Unison [6]. We use a similar system called the Syxaw XML-aware file synchronizer [7].

II. System design

Dessy was originally designed to bring a synchronization-aware desktop search mechanism to mobile devices. The design was prototyped in 2007 and results were presented in *Dessy: towards flexible mobile search* [8]. The system was later ported to MIDP [9]. The Dessy desktop search system was designed in Java with portability in mind. The structure of Dessy follows a modular design, as shown in Figure 1. File indexing, querying, metadata storage and synchronization are clearly separated. The Figure is color–coded. Yellow denotes components of the query engine. These translate the user's queries to retrieval requests to the index and synchronization requests to Syxaw. Every path is treated as a query in Dessy, so it is natural to handle browsing of both physical and virtual directories with the query engine. Green denotes the indexing engine. It retrieves document identifiers from the index implementation based on keywords. The indexing engine also contains the interfaces for metadata sources. These allow searching for files on the local filesystem, remote computers, and Internet servers. The red color indicates the Syxaw file synchronizer.

Synchronization, conflict reconciliation and associating files with metadata in Dessy is done by the Syxaw file synchronizer. Syxaw associates file data and metadata with a universal identifier (UID) that Dessy then uses to identify the file. Synchronization requests are passed to Syxaw in UID form. For remote hosts and the Internet, host names or IP addresses are used as prefixes of the UID to obtain *Globally Unique Identifiers* or GUIDs. These are used to synchronize a group of files with their original hosts, regardless of which files come from which host. The synchronization protocol uses HTTP requests initiated by the mobile client. To minimize the impact of the high latency of current cellular data networks,

[1] http://www.apple.com/macosx/features/spotlight/
[2] http://projects.gnome.org/tracker/
[3] http://www.copernic.com
[4] http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html
[5] http://jcp.org/aboutJava/communityprocess/final/jsr139

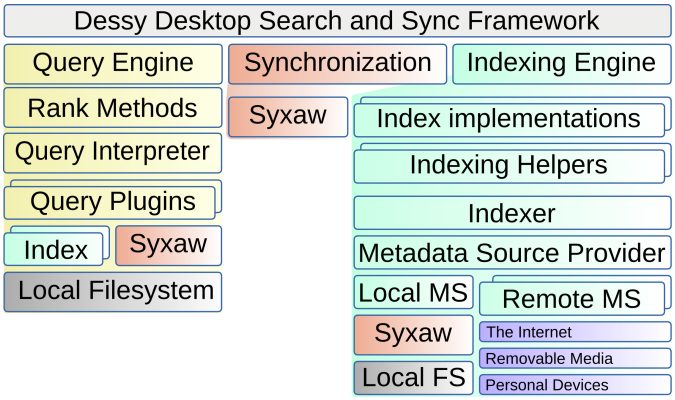

Fig. 1. A structural breakdown of Dessy.

we designed the protocol to make as few requests as possible. In particular, we batch object requests, so that a complete file system is typically synchronized in only two HTTP requests. To support varying pricing and hot-spots, data synchronization consists of two stages of network-intensive discrete runs, where the first stage synchronizes file metadata (i.e. the layout of the directory tree) and the second stage synchronizes file content. We contrast this to systems that impose a continuous, typically lighter, load on the network [10]–[12]. The Syxaw file synchronizer supports synchronizing individual files, sets of files, and whole file systems.

The directory tree or filesystem assigned to Dessy is crawled by an indexer that detects file types and invokes indexing helpers. These read their supported file types and generate summaries that are then stored into the index. To add support for a new file type, a developer needs to write an indexing helper for that file type. For example, PdfHelper reads PDF files for metadata and document text, and adds these as properties such as `author:`, `keywords:`, `title:`, and `text:`. Dessy stores the indexed properties in a lightweight key–value database based on the Sdbm in–file database.

III. Evaluation

The qualities of a desktop search system most visible to the user relate to the speed and ease of use of common operations of the system, and the system's ability to find desired documents. Since synchronization and file searching are very user-oriented tasks, we choose to evaluate the system from the user's perspective. We assume that most users are comfortable with a wait time of one second between starting a search and receiving results, and set this as a goal for Dessy. The other common actions in Dessy are synchronizing files, adding file tags, deleting files, and synchronizing file metadata. Adding new tags and deleting files are fast operations, since they require a single memory or disk access: adding a word to the index, and removing word identifier, respectively. Therefore, the operations that must be examined closer are search and synchronization of file data. The synchronization time and search time measurements were part of a Master's thesis on Dessy [9]. Therefore, we will only summarize them here, and concentrate on the energy usage measurements of Dessy. To measure the impact of Dessy on the client device, we measure the energy usage of Dessy on the Nokia E51 with several different usage patterns. The energy use is calculated from power measurements taken with the Nokia Energy Profiler[6].

To evaluate synchronization speed, we used a data set made of random UK English words. We picked the words from the *UK English Wordlist With Frequency Classification*[7] collection, designed for spellcheckers. We created files with 20 to 900 000 randomly chosen words. The final collection had 57 files ranging from 188 bytes to 5.2 megabytes.

The dataset was synchronized on the Sun WTK emulator and the Nokia E51, both connected to the synchronization server via an IEEE 802.11 wireless link. The details of the experiment were presented in the Master's thesis [9]. In summary, the time to synchronize seems to grow linearly from above 20 seconds on the E51, and from around a second on the emulator, to around 40 seconds. We believe the large performance gap is due to a bug, possibly to do with the way the JSR 75 implementation on the E51 handles files. Based on the measurements, it would appear that synchronization of larger files is both feasible and reasonably fast on the Nokia E51. However, the user still needs to wait for the synchronization to complete. This suggests that more work should be done on scheduled and background synchronization.

We also measured the time it takes to search for files remotely, and receive results on the Nokia E51. In this experiment, the phone connected to a laptop running the Dessy remote API, and requested files with random queries generated from the UK English words dataset introduced earlier. The number of query words was normally distributed, with the mean being 2.3 query words, as observed in mobile search studies [13], [14]. The standard deviation was 1. Each experiment run consisted of one thousand queries. We ran the experiment five times. The results were shown in the Master's thesis [9]. The searches completed on average in under half a second, and all completed in under 900 milliseconds. This well achieves the goal of searching in under a second. Furthermore, the search time of one second is extremely short compared to the average time that users spend inputting the queries. In the Google mobile search study [14], users took $56 - 63$ seconds on average to type a query to the mobile version of Google.

Next, we compare the energy efficiency of synchronization with different communication technologies: IEEE 802.11 wireless, 2G (EDGE) and 3G. Files on HTTP servers on the Internet and the same files on a Dessy server on a remote desktop machine are synchronized. We expect no differences between Internet and Dessy Server synchronization, since the client device is initially empty in this experiment, and compression of file data is disabled. This makes internet download and synchronization with a full download essentially the same operation. If the client had a previous version of

[6]http://www.forum.nokia.com/Tools_Docs_and_Code/Tools/Plug-ins/Enablers/Nokia_Energy_Profiler/

[7]http://http://www.bckelk.ukfsn.org/words/wlist.zip [downloaded 2009-05-27]

Experiment	Lifetime (h)	σ
NEP Idle	102.734800	11.506324
System Idle	60.779333	4.943411
Server WLAN	7.955036	0.409574
Internet WLAN	7.584398	0.126252
Internet 2G	7.988534	0.328137
Server 3G	4.347392	0.154648
Internet 3G	4.417638	0.219168

TABLE I
BATTERY LIFETIME IN SYNCHRONIZATION EXPERIMENTS.

Operation	Battery capacity	Average power
System startup	395 times	0.43 W
Initial connection	743 times	0.49 W
WLAN Synchronization	980 times	0.52 W
User interaction	15 hours	0.26 W
System idle	61 hours	0.069 W

TABLE II
BATTERY CAPACITY IN TERMS OF SYSTEM OPERATIONS.

the files, the transmission of differences instead of complete files would decrease transmission times and work in favor of synchronization with the desktop server.

Table I shows the battery lifetime of the E51 in different synchronization experiments. The lifetime is shown in hours with standard deviation (σ). We examine the battery life as a measure of energy efficiency between different scenarios. For comparison, we show the energy usage of the Nokia Energy Profiler when idle (*NEP Idle*) and that of Dessy and the NEP when idle (*System Idle*). We can see that the full Nokia E51 battery can handle Dessy for around 61 hours. The synchronization experiments shown in the table are: *Server WLAN* – The Dessy client on the E51 is sychronizing files through an IEEE 802.11 wireless LAN connection with the Dessy server running on a desktop machine. *Internet WLAN* – same as the former, except synchronization happens with different file servers hosting individual files on the Internet. In *Internet 2G*, the phone's 2G (EDGE) connection is used instead. In *Server 3G*, we synchronize with the Dessy server using the mobile phone's 3G network connection. Finally, *Internet 3G* means that the client is synchronizing files from Internet servers through a 3G connection.

Both Server WLAN and Internet WLAN have a higher battery lifetime than 3G. However, 2G is in par with WLAN in energy efficiency. This leads us to believe that the IEEE 802.11 connection is preferable to 2G when high speed or cost–free synchronization is required. However, when network hotspots are scarce, 2G is preferable to 3G, since it is more energy efficient. The differences between synchronizing with a Dessy server and the Internet servers are minimal.

Next, we examine the power usage profile of Dessy operations. Table II summarizes the energy usage measurements of different Dessy operations with numbers and durations permitted by a full standard battery of the Nokia E51 on the device. `WLAN Synchronization` refers to using the Dessy software to synchronize a PDF file 0.5 MB in size.

IV. CONCLUSION

The results in the previous section suggest that desktop search is both feasible and searches complete quickly on average mobile devices.

Synchronization of files could be quicker, however. The performance gap between the emulator and the Nokia E51 in the synchronization experiments remains to be solved.

The measurements suggest that desktop search and synchronization can be implemented in an energy efficient way on mobile devices. The energy usage of Dessy is around 0.26 W when active and 0.5 W when transmitting or receiving data. The power usage when idle is 0.07 W, giving a standby time of 61 hours for Dessy. For comparison, the phone uses 0.04 W when idle without running Dessy, and reaches 102 hours of standby time. The power usage of synchronization on a 3G network is much higher than that of IEEE 802.11 wireless. This supports the conclusion that synchronization at a network hotspot is preferable to 3G. Based on this work, it is possible to develop systems that maximize battery life while providing useful search and synchronization capabilities.

REFERENCES

[1] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. James W. O'Toole, "Semantic file systems," in *SOSP '91*, pp. 16–25.

[2] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin, "Fast, flexible filtering with phlat," in *SIGCHI '06*, pp. 261–270.

[3] H. Mei and J. Lukkien, "A remote personal device management framework based on syncml dm specifications," in *MDM '05*, pp. 185–191.

[4] T.-Y. Chang, A. Velayutham, and R. Sivakumar, "Mimic: raw activity shipping for file synchronization in mobile file systems," in *Mobisys 2004 Workshop on Context Awareness*, pp. 165–176.

[5] *SyncML Sync Protocol, version 1.1*, SyncML Initiative, Feb. 2002. [Online]. Available: http://www.syncml.org/docs/syncml_sync_protocol_v11_20020215.pdf

[6] E. I. Bolso, "File synchronization with unison," *Linux Journal*, vol. 2005, no. 132, pp. 6–6, 2005.

[7] T. Lindholm, J. Kangasharju, and S. Tarkoma, "Syxaw: Data synchronization middleware for the mobile web," *Mobile Networks and Applications*, vol. 14, no. 5, pp. 661–676, 2009. [Online]. Available: http://dx.doi.org/10.1007/s11036-008-0146-1

[8] E. Lagerspetz, T. Lindholm, and S. Tarkoma, "Dessy: Towards flexible mobile desktop search," in *Proceedings of the Fourth ACM SIGACT-SIGOPS International Workshop on Foundations of Mobile Computing*.

[9] E. Lagerspetz, "Dessy: desktop search and synchronization," Master's thesis, University of Helsinki, Department of Computer Science, Helsinki, Finland, Nov. 2009. [Online]. Available: http://www.cs.helsinki.fi/u/lagerspe/publications/lagerspe-gradu.pdf

[10] K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and A. J. Demers, "Flexible update propagation for weakly consistent replication," in *Proceedings of the sixteenth ACM Symposium on Operating Systems Principles*, pp. 288–301.

[11] J. Kubiatowicz *et al.*, "Oceanstore: An architecture for global-scale persistent storage," in *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems*.

[12] L. Mummert, M. Ebling, and M. Satyanarayanan, "Exploiting weak connectivity for mobile file access," in *Proceedings of the Fifteenth ACM Symposium on Operating System Principles*, pp. 143–155.

[13] J. Yi, F. Maghoul, and J. Pedersen, "Deciphering mobile search patterns: a study of yahoo! mobile search queries," in *The Seventeenth World Wide Web Conference*, pp. 257–266.

[14] M. Kamvar and S. Baluja, "A large scale study of wireless search behavior: Google mobile search," in *SIGCHI '06*, pp. 701–709.