Pocket Backup Storage System with Cloud Integration

Karthik Nilakant and Eiko Yoneki
University of Cambridge, Computer Laboratory
Cambridge CB3 0FD, United Kingdom
{karthik.nilakant}{eiko.yoneki}@cl.cam.ac.uk

ABSTRACT

A variety of personal backup services now allow users to synchronise their files across multiple devices such as laptops and smartphones. These applications typically operate by synchronising each device with a centralised storage service across the Internet. However, access to the Internet may occasionally not be available, leaving any unsynchronised content in a vulnerable state. To address this, we introduce a cooperative backup storage system that could alternatively make use of storage capacity provided by other devices within close proximity, using ad-hoc or local network connectivity. Such devices can provide a secondary storage tier in case of Internet connectivity issues, and could also be used to forward files to central storage at a later time.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer Communication Networks – Distributed Systems

General Terms

Measurement, Experimentation

Keywords

Pocket Switched Networks, Storage System

1. INTRODUCTION

Centralised, hosted file storage services are now widely used, and for many users have begun to supplant external storage devices as a means of backing up personal documents and other files. These services enable file synchronisation amongst multiple devices, allow documents to be accessed from any Internet-connected machine, and facilitate sharing and collaboration of stored content. Typically, the services utilise large clusters of servers to facilitate high availability and low latency. Popular examples include Google Drive, Dropbox, Microsoft's SkyDrive and Apple's iCloud.

However, there are a range of special circumstances where lack of connectivity restricts the functionality of these services. For example, a group of tourists may be taking photographs on their smartphones while on vacation. One person's photographs may be initially stored on flash memory within the phone, and synchronised with a centralised storage service such as those described above. However, while the tourists are "roaming", data access may not be available or may be prohibitively expensive. If the tourists lose or damage their phones, then all of their unsynchronised photographs will also be lost. This could be avoided if the smartphones were able to cooperatively back up locally stored files, using peer-to-peer wireless networking. Such networking capabilities are available on most mobile smartphone platforms today.

Although it may be possible to construct decentralised storage services that operate using opportunistic short-range communication between devices, the costs are often thought to outweigh any possible benefits of such an approach. Wireless Internet connectivity is widely available throughout the developed world, through WiFi access points and cellular data networks. Conversely, opportunistic routing in networks of mobile agents is an inefficient process usually requiring redundant replication of data bundles.

In our proposed solution, a centralised service should be utilised whenever possible due to the increased efficiency and reliability of that approach. However, when access to the centralised service is unavailable, the backup service could temporarily switch to a decentralised mode of operation. If "cloud storage" refers to centralised services accessed over the Internet, one might coin the term "mist storage" to refer to these temporary decentralised storage systems. We introduce the design and operation of a mist-based cooperative backup service called "Mistify", which takes advantage of the data storage capabilities of other participating clients in a local area, by distributing encrypted content for safekeeping.

2. SYSTEM DESIGN

Mistify makes use of the Haggle framework for opportunistic communication [1]. Haggle provides two key features. Firstly, its content-centric "search based" API allows Mistify to interact with a virtual content repository, by publishing tagged content, and subscribing to those tags. Secondly, Haggle arranges for opportunistic, delay-tolerant forwarding of content between publishers and subscribers, using any available networking interface such as WiFi or Bluetooth.

2.1 Strategy

By making use of Haggle, Mistify behaves as a topology independent system, and does not track other devices on the network; instead, it discovers the attributes of content held in the mist. It then develops interests in specific content, based on relationships between the user and the content's owners. Mistify's replication strategy attempts to minimise over-replication of data, and prioritises certain content based on social relationships between users and their files.

CHANTS'12, August 22, 2012, Istanbul, Turkey.
ACM 978-1-4503-1284-4/12/08.

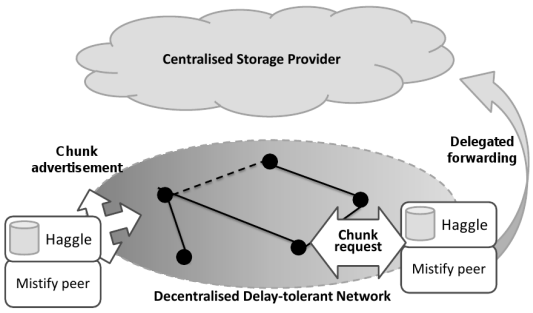

Figure 1: Diagram showing a conceptual overview of the Mistify architecture. The mist service, implemented in a virtualised testbed, is arbitrated by the Haggle client on each peer. Mistify also utilises cloud storage (implemented using an Amazon EC2 instance) if available.

Mistify has two basic constructs that are mapped to Haggle's "data objects": "Chunks" and "Entitlements". Chunks are bundles of encrypted data and associated metadata, which correspond to files within a user's replicated filesystem. A Chunk's owner is the original user that published the Chunk. It is envisaged that Chunks could also be used to aggregate smaller files, or disaggregate larger files into manageable bundles. Entitlements are used to advertise Chunks for retrieval by other peers. Private entitlements allow groups of users (other than the content owner) to access a bundle of encrypted Chunks, whereas public entitlements are used to advertise Chunks for replication to other possibly untrusted peers. The architecture is illustrated in figure 1. To assess the safety of a Chunk, clients check the public Entitlements published by other peers in the network. All Chunks are encrypted before publication. Private entitlements can be used to grant access to this encrypted content.

Replication is a two step process. Each peer subscribes to public Entitlements that are published to the mist by other peers, resulting in this low-volume metadata being flooded through the network (similar to routing updates in a link state routing protocol). Each Entitlement contains one or more "seeds", that advertises Chunks that are available for retrieval from that user. In order to subsequently retrieve a Chunk that has been advertised, Mistify registers an explicit interest in that Chunk with Haggle. Haggle will then arrange for the Chunk and its encrypted content to be forwarded to the requester. Whenever a peer retrieves a recent Entitlement, this has the effect of refreshing the availability of those seeds. Each Mistify peer makes an independent assessment of the safety of each Chunk, by keeping track of the number of replicas advertised by other users in the same locality.

2.2 Safety calculation

In general terms, a Chunk's "safety" is an estimate of the likelihood that a replica can be obtained from the mist within a pre-defined time interval. Since this calculation depends on a number of variables, these are combined into a single formula:

$$S = \frac{1}{R} \sum \frac{T - t_p}{T}$$

Where T is the maximum time threshold to regard a peer as available, t_p is the amount of elapsed time since a public Entitlement containing the Chunk was received from peer p, and R is the targeted number of replicas for each Chunk. Only peers that have been contacted within the time window specified by T are considered for each Chunk. After performing this calculation on each Chunk, those with a safety value exceeding 1 are regarded as safe (that is, enough replica advertisements within the safety window have been witnessed in the repository).

3. RUNNING MISTIFY IN HAGGLE

The trials were performed using a Haggle testbed [1], which is built on a PC with an 8-core CPU and 24GB RAM running Debian Squeeze, and installed with the Xen hypervisor. An array of virtual nodes was instantiated on the testbed system. Each of the nodes run a basic Debian-based operating system, and also feature a Java runtime environment (for running Mistify). The virtual nodes are each allocated with 128 megabytes of RAM, and a 1 gigabyte local disk. To drive each trial, a test scenario runner initialises the environment, starts the application on each node, and models changes in topology by analysing events in a pre-collected network trace [2]. The original trace files contain records of connectivity between nodes, which can be translated into scripts by the scenario runner that block or allow traffic between the corresponding virtual nodes over the virtual network bridge.

The connectivity trace was used to activate firewall rules on the host for the testbed's virtual network. For instance, when a period of connectivity between two nodes began, a corresponding command would be executed on the host, enabling network traffic between the matching virtual instances in the testbed. The trace data from the study spans several days, however for this trial a period of three consecutive days from the trace was used. Time was accelerated in the trials, such that one day of trace activity passed in approximately five minutes. This has the effect of reducing intra-contact time, which means that data throttling rates must be set aggressively to allow effective replication. The participants had varying characteristics in the contact network. We define "aggregate degree" as the total number of unique nodes that were in contact with a particular node throughout the period, and "average degree" as the mean number of contacts that a node had at each time interval.

4. CONCLUSIONS

In our evaluation of the prototype, we have found that in a simulated network of locally-connected peers, the prototype was able to achieve a high level of availability for stored content, without resorting to flooding. Furthermore, Mistify was able to deliver a high proportion of content to the cloud, even when only a small proportion of nodes were given Internet connectivity.

Acknowledgement. This research is part-funded by the EPSRC DDEPI Project, EP/H003959.

5. REFERENCES

[1] Haggle. http://www.haggleproject.org.
[2] P. Hui, J. Crowcroft, and E. Yoneki. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In *MobiHoc*, 2008.