Mistify: Augmenting Cloud Storage With Delay-Tolerant Cooperative Backup

Karthik Nilakant, Jon Crowcroft and Eiko Yoneki
University of Cambridge Computer Laboratory
Cambridge, United Kingdom
Email: firstname.lastname@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, applications 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.**

In our proposed design, we delegate the task of propagating information across locally networked devices to a lower layer, by making use of a content-centric opportunistic network platform (Haggle). This allows our application, Mistify, to treat the neighbourhood of peers as a single distributed content repository (or "mist"), in a manner similar to the way in which existing applications interface with the cloud. Mistify employs a differentiated replication strategy, with the aim of improving the safety of items in the mist. 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.

I. Introduction

Centralised, hosted file storage services such as Google Drive, Dropbox, Microsoft's SkyDrive and Apple's iCloud 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.

There are, however, a range of circumstances where lack of connectivity restricts the applicability of these services. To illustrate: 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. Commonly, while cellular phones 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. However, 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". Mistify takes advantage of the data storage capabilities of other participating clients in a local area, by distributing encrypted content for safekeeping.

The primary contributions of this paper are:

1) Demonstrating that an effective second-tier storage service can be built using an existing framework for delay tolerant, opportunistic networking. We evaluate a working prototype on a virtualised network testbed, using human interaction data gathered from a real mobility experiment.
2) Illustrating how this decentralised mode of operation can be integrated with a simple centralised storage model, providing a supplementary level of service.

Our evaluation of the prototype showed that even peers that were in the lower quartile of the network's degree distribution could achieve successful replication for at least 70% of their files, and in most cases above 90% coverage is possible for well-connected peers. Furthermore, through delegated forwarding, approximately 30% of the community's content can be transferred to central storage even when less than 20% of peers have Internet connectivity.

II. Architecture

Mistify aims to satisfy two basic requirements. Firstly, content of specific interest to a user needs to be retrieved as a matter of priority. This includes content that is either owned by the user (typically retrieved during a restore operation), or has been explicitly entitled to the user. Secondly, Mistify peers attempt to boost the safety of any content in the mist, by retrieving it for local storage, if it is deemed necessary (this process is called "locality replication"). In order to achieve these aims in an environment that is subject to on-going change, Mistify employs delay-tolerant opportunistic networking.

A. Opportunistic networking

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.

Delay tolerance is provided through use of the PROPHET protocol [2]. PROPHET is currently an IETF draft, and operates by recording histories of contact with other peers. The probability of being in contact with each other peer is derived from this history, and a vector containing these probabilities is distributed to other peers. PROPHET can then calculate transitive probabilities for forwarding content, via peers that are closer probabilistically.

B. 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.

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.

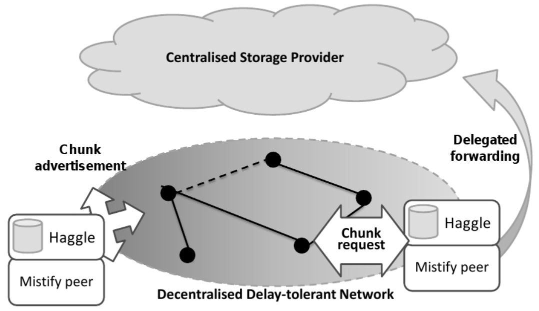

Figure 1: Diagram showing a conceptual overview of the Mistify architecture. The mist service, implemented in a virtualised testbed, is arbitrated by the Haggle client on each peer. Mistify also utilises cloud storage (implemented using an Amazon EC2 instance) if available.

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.

III. Implementation

In this section, we describe the core components and tuning parameters that make up the Mistify prototype. The prototype is a Java application, which connects to the Haggle framework using the *libhaggle* API library over the Java Native Interface. Haggle is a native C++ application, which can be compiled on multiple platforms such as Linux, Windows and Android smartphones. In this particular study, the performance of the prototype was evaluated using a virtualised testbed environment, which is described further below.

A. 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). Each of these parameters are configurable at run-time, and are summarised in table I.

B. Strategy refinements

In addition to the basic safety preservation strategy described above, two further refinements were required to cater to specific scenarios.

1) Containment: Chunks could be replicated to a device that has roamed outside its "home" community. When the device returns to its home community, the safety metric of the replicated Chunk within the remote community will drop, causing the Chunk to be replicated in a network that the original content owner cannot access. To counteract this, the selection process was modified to ensure "owner presence". To assess this, each seed that is a candidate for retrieval must have a current replica that is advertised by the content owner, within a time interval denoted by an owner presence factor' (this parameter is expressed as a proportion of the safety window).

2) Request collision: Another issue that leads to over-replication is that when a new node enters a network of previously connected nodes, the existing nodes will learn about new content at approximately the same time, and will try to replicate it simultaneously. To avoid this, the list of unsafe Chunks is shuffled by each peer prior to selection and random offsets are added to each Chunk selection interval. The throttling rate is also scaled by the number of local peers currently detected, so that replication will proceed more slowly in larger co-located groups.

C. Cloud integration

In the current prototype, an SFTP service is used to model a passive central storage provider. Mistify connects to the cloud service using a separate module that regularly checks for connectivity and if available, initiates synchronisation. Mistify also allows forwarding to the cloud via other (possibly untrusted) peers. To facilitate this, each user allows public access to a known location in their cloud repository. This is currently implemented using a special directory in each user's SFTP path. Each Chunk contains the necessary metadata to allow other peers to locate the owner's cloud repository. Peers upload Chunk content into this repository, and leave a serialised Entitlement to allow the owner to identify the uploaded Chunk. When one of the owner's devices has connectivity to the cloud, it will process the contents of the public directory, and move the associated files to the appropriate locations in the main repository.

D. Encryption

In the current prototype, a lightweight encryption scheme has been used. The main purpose of this is to show how cryptography fits into the framework. A third-party library known as "Jasypt" provides the bulk of the cryptographic functionality. If a user subsequently wants to entitle some Chunks to a group, an Entitlement is created, where each associated Chunk's key is re-encrypted with the target group's key.

E. Haggle testbed

The trials were performed using a Haggle testbed, 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. 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.

IV. Evaluation

In order to evaluate the operation of the prototype under realistic conditions, the testbed was driven by connectivity trace data collected from a prior study [3]. In that study, a group of 35 undergraduate students were given tracking devices, which recorded contacts with other devices in close range. In order to simulate this activity in the testbed environment, the connectivity trace was used to activate firewall rules on the host for the testbed's virtual network.

Parameter	*Description*	*Default*
Advertisement rate	The application will publish public Entitlements at a rate specified by this parameter.	15 seconds
Chunk safety window length	Only Chunk replicas identified in public entitlements retrieved within this time interval are considered in the safety calculation.	300 seconds
Target number of replicas	Allows Mistify to attempt to maintain an increased number of (explicit) replicas per Chunk.	1 replica
Owner presence factor	In order to consider a Chunk for local retrieval, an Entitlement containing the Chunk must be received from the Chunk owner, within the time interval specified by this parameter. The value is expressed as a multiple of the safety window length.	0.5
Maximum number of tracked peers	When calculating Chunk safety, the application will only use Entitlements from a number of recently contacted peers, as specified by this parameter.	40
Locality replication throttling rate	The application is able to request unsafe Chunks at a rate not exceeding this parameter (scaled by the number of peers in the locality).	2 MiB / minute

Table I: Description of available parameters used to tune Mistify's strategy subsystem.

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.

A. Mist mode operation

In the first set of trials, Internet connectivity was completely disabled, forcing Mistify to make exclusive use of decentralised local storage. To initialise the scenario, one of the virtual instances was selected as a content owner, and was populated with ten megabytes in 100 files. The remaining participants were initialised with separate user IDs, and would make use of locality replication to improve the safety of the owner's Chunks. For each of the trials, traffic on the virtual network was recorded, in addition to various metrics as seen on the content owner. The final state of the content caches on each node was also recorded at the conclusion of each trial. Unless otherwise stated, the default tuning parameters as shown in table I were used in each trial. In the following subsections, we discuss the effect of varying each of the parameters in this scenario. The results of each trial are summarised in table II. The results from this part of the evaluation could be used to design specific policies based on resource availability and user preference, which are implemented using the tuning parameters described above.

1) Base case: In our first trial, the prototype was configured with all parameters set to default values. A node with approximately median characteristics was chosen as the content owner (node 28, which was ranked 15th for aggregate degree, and 21st for average degree). In table II, the results after adjusting various parameters described in the following sections is shown. "Availability" indicates the proportion of Chunks that were replicated; "upper quartile availability" shows the proportion of Chunks that were replicated to the nodes closest neighbours; "replicas" shows the total amount of Chunk replication; "explicit requests" identifies the proportion of those replicas that were retrieved as a result of an explicit request by the application. In an ideal scenario, all Chunks would be replicated to a small group of commonly contacted peers, and the number of replicas created for each Chunk would be minimised. In figure 2, the aggregate data rate of traffic over the duration of the trial has been plotted, in order to show the level of activity. Overlaid on the traffic graph are plots of medium-term and short-term availability, as observed by the content owner. This shows the proportion of Chunks that the owner had seen on the network within the default safety window. Note that these figures are measured at the application layer, and therefore exclude implicitly replicated Chunks. In figure

Configuration	*Availability*	*Top 25%*	*Replicas*	*Explicit*
Base case	0.97	0.94	1041	0.18
see §*IV-A2*				
Node 34	0.92	0.90	782	0.14
Node 16	0.99	0.99	1077	0.12
Node 26	0.72	0.70	443	0.18
see §*IV-A3*				
30s window	0.84	0.84	342	0.09
60s window	0.86	0.85	577	0.20
10m window	0.99	0.99	937	0.14
see §*IV-A3*				
No presence	0.99	0.98	1133	0.10
0.1 Presence	0.79	0.75	506	0.12
0.25 Presence	0.96	0.81	890	0.21
1.0 Presence	0.93	0.85	924	0.15
see §*IV-A3*				
5 peers	0.86	0.84	827	0.09
15 peers	0.92	0.92	897	0.17
see §*IV-A4*				
1 MB/min	0.56	0.46	216	0.23
4 MB/min	1.00	1.00	1186	0.21

Table II: Final state of peers.

3 (left), the distribution of files is shown, with respect to the number of replicas that were found in the network. Separate curves were plotted for explicit and implicit replication – that is, replicas that were created due to a Chunk request generated by Mistify, or replicas generated by Haggle's delay-tolerant forwarding subsystem (note that each explicit replica is also counted as an implicit replica, since it will occupy the content cache at the requesting node). A mean of 10.4 implicit replicas per Chunk were created. Conversely, figure 3 (right) shows the cumulative frequency distribution of the combined (implicit) Chunk replicas across the nodes in the environment. The surge towards the right hand side of the curve shows that there are a concentration of nodes that receive substantially more replicas as a result of delay-tolerant routing, which is also reflected in the upper quartile availability figure.

2) Owner characteristics: The delay-tolerant platform should allow any type of node to replicate its content amongst the other nodes in the environment, possibly with a longer propagation delay for those nodes that are not well connected in the network. To evaluate this, a selection of nodes with specific characteristics were selected as originators of content in separate trials. Node 34 had a high average degree but only a median aggregate degree. Node 16 had a high aggregate degree but a median average degree. Node 26 was ranked in the bottom quartile for both measures. In each graph, values for medium-term availability (replicas seen within the safety window), and short term availability (replicas seen within the time taken to publish two Entitlements) are plotted, as observed by the Mistify instance on the owner node. The application is only aware of Chunks that have been explicitly replicated to other nodes in the owner's locality, which means the plotted figures will underestimate Chunk availability. As one might expect, nodes with a higher level of connectivity in the network tend to see more replicas in their immediate locality. Table II summarises the final distribution of both implicit and explicit replicas in the network. In each case, over 70% of the owner's files were successfully replicated to other nodes in the mist. Node 26 replicated fewer Chunks, but still maintained a high rate of coverage. The figures for implicit replication for all types of nodes suggest that if enough time is allowed for retrieval over the DTN, even nodes with lower than average connectivity characteristics should be able to restore the majority of their Chunks if the need arises.

3) Safety evaluation: As described earlier, only Chunks that a Mistify peer has seen advertised within a pre-defined safety window are considered for local retrieval. Furthermore, the "owner presence" criterion requires that one of the Chunk advertisements must originate from the owner of that content, within a time period specified by the presence factor parameter. The default safety window length of five minutes roughly equates to 24 hours of accelerated time in the trace data. An analysis of the connectivity trace for node 28 revealed a mean inter-contact interval of approximately 3.6 hours, or 45 seconds in the testbed environment. In the next set of trials, the window size was set to 30, 60 and 600 seconds, holding all other parameters at their default values. Interestingly, an order-of-magnitude decrease in window length (from 600 to 60 seconds) only resulted in a marginal decrease in Chunk availability (from 99% to 85%), although it markedly reduced the number of replicas that were produced. This result seems counter-intuitive, since a longer safety window implies that a peer will avoid retrieving Chunks with existing replicas for longer. To investigate this further, the owner presence factor was also tested. The trials were repeated using values of 0, 0.1 0.25 and 1.0 for the presence factor (note that a value of zero has the effect of disabling the owner presence check). The results appear to show a similar pattern to the safety window figures, however it seems that the act of enabling the presence check is enough to significantly stem over-replication.

In the previous trials, the prototype was configured to collect Entitlements from every other peer in the network. In a network of the size used in these experiments, the computational overhead imposed by tracking all peers is unlikely to be significant. However, the tests were repeated with the peer tracking limit set to 5 and 15 peers, to highlight any differences in behaviour. The results do not show a dramatic increase in replication – this is probably due to the fact that many peers are already filtered out of the safety calculation by the safety window and owner presence criteria.

4) Throttling: The throttling rate is an upper limit on the total amount of data in Chunks that a Mistify peer may explicitly retrieve per time unit. A high rate may result in over-replication of Chunks, while a low rate may result in poor recoverability of Chunks. The trials were repeated with the throttling rate set to 1000 and 4000 kilobytes per minute (in accelerated time). Again, the results are presented in table II. The results show an expected increase in Chunk availability and explicit / implicit replication. Unfortunately, the successive increases in replication coverage are relatively minor compared to the amount of over-replication that occurs.

B. Mixed mode operation

In the next scenario, the aim was to evaluate the effect of introducing connectivity to a centralised storage repository. There are two specific use cases that are demonstrated in this section: firstly, the behaviour of the application when Internet connectivity is restored at a specific time in the trial; and the behaviour of the application when a subset of peers are granted Internet access for the duration of the trial. All tuning parameters were set to default values, and in these trials all nodes were populated with one megabyte of data in ten files, in order to mitigate any bias caused by selecting a specific node as a content originator.

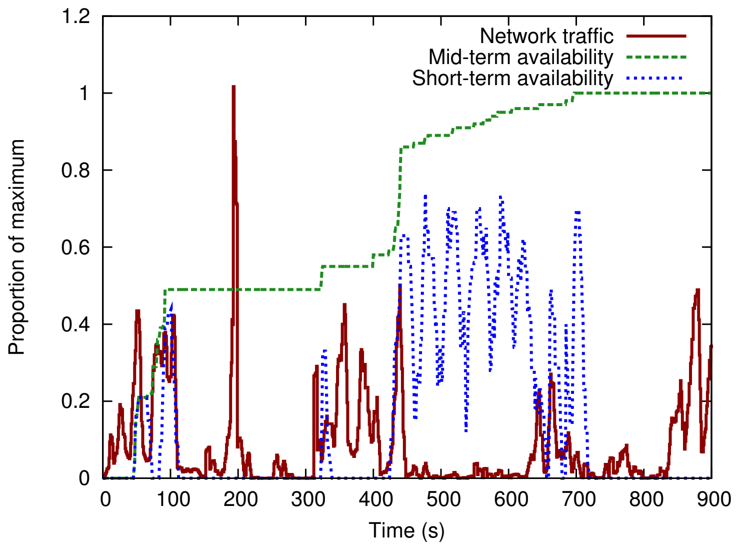

Figure 2: This graph shows the level of network activity in the virtual network, together with a local view of replica availability from the perspective of the content owner. All parameters were set to default values for this trial.

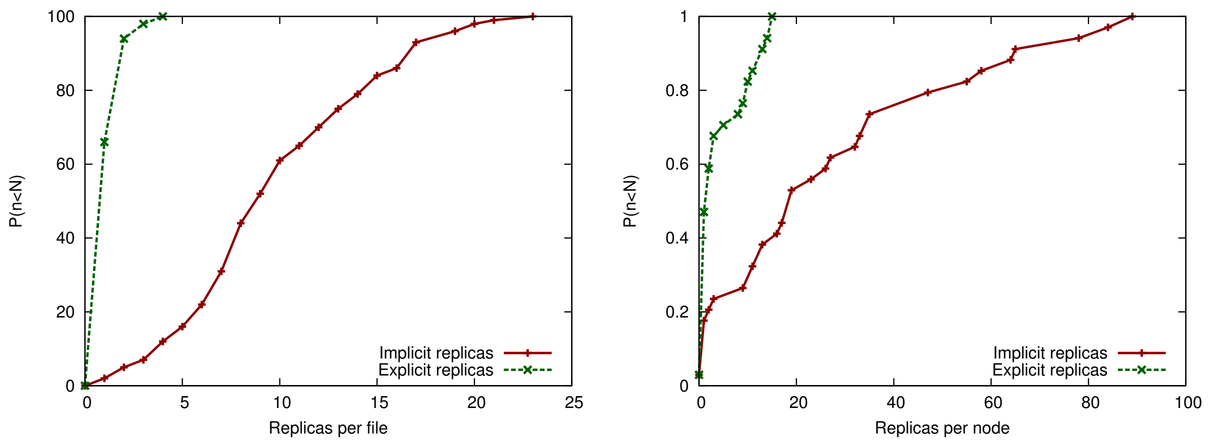

Figure 3: These cumulative distribution functions show how replicas were spread through the network in the base case trial. The left graph shows the number of replicas seen per file, whereas the right graph shows the number of replicas stored at each peer.

1) Cloud connectivity response: As shown above, utilising the mist carries the cost of Chunk over-replication. A desirable property would be to ensure traffic in the mist ceases as soon as Chunks are able to be replicated to the cloud over an Internet link. However, just as Chunk advertisements must propagate through the DTN to allow content to be replicated, a similar advertisement must instruct peers to stop replicating. To evaluate this response, a trial was conducted where Internet connectivity was enabled on the content originator at a specific time in the trace. Note that delegated forwarding to the cloud was disabled in this case. Specifically, the chosen connection time occurred at 50% through the duration of the scenario. Figure 4 illustrates network traffic seen in the virtual network over time, in each of the trial cases. The graph shows that the surge in replication traffic towards the end of the trace is avoided when Internet connectivity is available – however, there is still a moderate level of replication traffic, most likely caused by earlier requests that have propagated through the network.

2) Delegated cloud backup: To evaluate the second mixed-mode use case, subsets of nodes were randomly selected from the network to act as delegated forwarders of Chunks. The trial was conducted using groups of five, ten and fifteen nodes with Internet connectivity enabled. Each trial was repeated with ten different randomly selected groups. The results are presented in table III. The table shows the mean proportion of Chunks that were successfully forwarded to the cloud repository in each trial configuration.

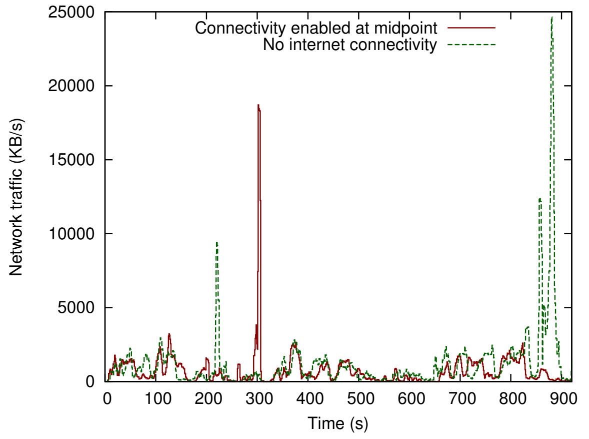

Figure 4: These two network traces illustrate how Mistify responds to the availability of the cloud. In the second trace, Internet connectivity was enabled at a point halfway through the trial.

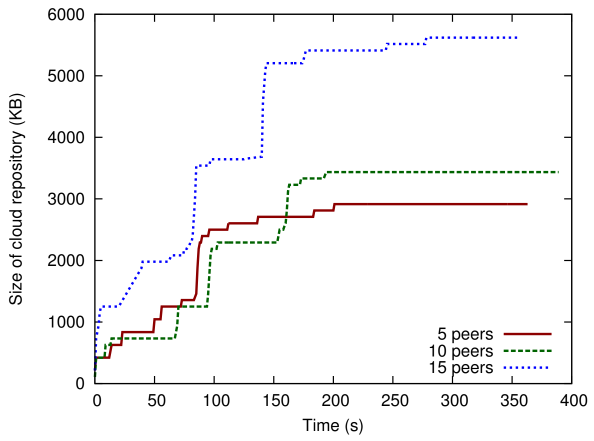

Figure 5: This graph plots the size of the cloud content repository as it changes over time, with subsets of different sizes connected to the cloud.

As discussed earlier, the current architecture means that Mistify is only able to forward Chunks that have been explicitly requested, since implicit replicas are hidden within the Haggle layer. If access to Haggle's content cache was possible through API hooks, a higher proportion of Chunks could be replicated to central storage. To get an indication of the improvement in availability that could be brought about by this, the prototype was modified to copy files directly from Haggle's content repository to the cloud (note that these files are devoid of metadata, and therefore would not be able to be restored by the owner). These figures are shown in the next column of the table, and indicate that if the layering constraints of the prototype were relaxed by allowing further API access to Haggle's internal workings, an increase of 2 to 2 times as much forwarded data may be possible. In figure 5, traffic traces from three typical trials in this scenario are shown, illustrating the improvement brought about with more Internet-connected peers.

Number of forwarders	*Uploaded replicas*
5	29.3
10	42.7
15	51.3
5 (modified)	76.3

Table III: This table shows the number of replicas that were successfully forwarded to the cloud by random groups of different sizes. The modified trial allowed implicitly replicated content to be uploaded.

V. Related work

The type of applications that are most similar to Mistify can be classed as "cooperative backup". Examples include Pastiche[4], Cimbiosys[5], MoSAIC[6] and PodBase[7]. Each of these applications make use of opportunistic storing-and-forwarding to provide a short-term archival service. Cimbiosys is a peer-to-peer filtered replication system, which has been adapted to integrate with delay-tolerant routing strategies. In that study, a vehicular network trace was used to evaluate the efficacy of a messaging system built on the prototype. MoSAIC also provides a similar service to Mistify, with the aim of eventually forwarding data to a central provider. The main difference is that Mistify can make use of the underlying delay-tolerant forwarding framework to map out more complex relationships between users and their data.

Other research projects similar to Mistify have been developed to assist in the evaluation of lower layer frameworks. For example the Anzere storage system makes use of a constraint-based distributed execution framework known as Rhizoma [8]. Anzere supports decentralised dissemination and collaborative storage of smartphone files such as images and videos. Conversely, the Courier smartphone-based system uses a host-centric mechanism to allow higher level applications to take advantage of storage capabilities offered on peer devices, through the service [9]. As an example, the authors developed an application that utilises the storage service to "swap out" files on a smartphone with a local storage repository, increasing the smartphone's "virtual" storage capacity in a similar manner to a memory paging system.

Another branch of distributed storage research has focused on improving consistency of updates to data. The Bayou database [10] is seen as a pioneering example of using weak consistency replication techniques to provide file synchronisation operation in a frequently partitioned network. The PRACTI framework followed on from that project, which defines the necessary requirements for an efficient "eventually consistent" storage system, utilising version vectors and partial replication. Mistify could be enhanced with PRACTI functionality, to allow storage of collaborative data. Erasure coding instead of replication is another possible option, however we must first establish whether or not an erasure coding approach is feasible in networks of this nature [11].

VI. Conclusions and Future Work

Mistify is able to provide a reasonable level of safety for content, in the absence of Internet connectivity and even for users with limited local connectivity. It is able to achieve a high level of file coverage without flooding the network with replicas. Finally, Mistify is also able to leverage cloud connectivity, even when only a fraction of participants in the community are Internet-connected. The results presented above will also provide the basis for a more adaptive tuning system, which modifies parameters such as the safety window and throttling rate at run time, in order to adapt to different types of network conditions.

As discussed above, providing API access into Haggle's content store will allow Mistify to learn about implicitly replicated Chunks that have accumulated due to delay tolerant forwarding. This will allow the application to react to this type of traffic and reduce over-replication. Another approach would be to modify the functionality of Haggle's interest objects, by allowing the application to specify a scope for each interest. This would allow Mistify to specify a general interest in Chunks from the local content repository, interests in local Chunks only from direct neighbours, and interests in remote Chunks via delegated forwarding. Extending this further, interests could be marked with a "time to live" field, to reduce the effect of poor Chunk selections that propagate widely through the network. Finally, the application will also need access to Haggle's resource management layer. This will allow it to throttle traffic flowing through the peer due to implicit replication, which is currently not able to be controlled by Mistify.

Further development of the application will require trace data from other studies. Of particular interest are contact traces which also record the availability of network infrastructure to each participant. This will allow us to conduct mixed-mode tests without needing to randomly select Internet-connected participants. The testbed environment will also need to be upgraded with further resources in order to be able to conduct larger-scale experiments with hundreds of nodes. These types of experiments will allow us to assess the scalability of the application (in terms of users and content), which needs to be ensured before any live user trials will be possible.

Acknowledgement

This research is part-funded by the EU grants for the RECOGNITION project (FP7-ICT 257756) and the EPSRC DDEPI Project, EP/H003959.

References

[1] J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton, "Haggle: seamless networking for mobile applications," in *Proceedings of the 9th international conference on Ubiquitous computing*, ser. UbiComp '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 391–408. [Online]. Available: http://dl.acm.org/citation.cfm?id=1771592.1771615

[2] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/961268.961272

[3] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks," in *MobiHoc*, 2008.

[4] L. P. Cox, C. D. Murray, and B. D. Noble, "Pastiche: making backup cheap and easy," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 285–298, December 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844155

[5] P. Gilbert, V. Ramasubramanian, P. Stuedi, and D. Terry, "Peer-to-peer data replication meets delay tolerant networking," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, june 2011, pp. 109 –120.

[6] L. Courtes, O. Hamouda, M. Kaaniche, M.-O. Killijian, and D. Powell, "Dependability evaluation of cooperative backup strategies for mobile devices," in *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, dec. 2007, pp. 139 –146.

[7] A. Post, P. Kuznetsov, and P. Druschel, "Podbase: transparent storage management for personal devices," in *Proceedings of the 7th international conference on Peer-to-peer systems*, ser. IPTPS'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 1–1. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855641.1855642

[8] Q. Yin, A. Schpbach, J. Cappos, A. Baumann, and T. Roscoe, "Rhizoma: A runtime for self-deploying, self-managing overlays," in *Middleware 2009*, ser. Lecture Notes in Computer Science, J. Bacon and B. Cooper, Eds. Springer Berlin / Heidelberg, 2009, vol. 5896, pp. 184–204.

[9] A. Karlson, G. Smith, B. Meyers, G. Robertson, M. Czerwinski, A. Karlson, G. Smith, B. Meyers, G. Robertson, and M. Czerwinski, "http://research.microsoft.com courier: A collaborative phone-based file exchange system," 2008.

[10] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch, "The bayou architecture: Support for data sharing among mobile users," in *IEEE WMCSA*, 1994.

[11] W. Lin, D. Chiu, and Y. Lee, "Erasure code replication revisited," in *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference on*, aug. 2004, pp. 90 – 97.