Persistent e-mail identification is viable!

Stefan Haun and Andreas Nürnberger

Data and Knowledge Engineering Group,
Faculty of Computer Science,
Otto-von-Guericke-University Magdeburg, Germany
http://www.dke.ovgu.de

Abstract. Persistent identification of entities in Personal Information Management (PIM) is necessary to enable stable, long-term references in archives and semantic applications. In the case of e-mails, the standard offers Message-IDs (MID), which are widely deployed. However, stores do not use the MID but rather rely on a path, which is likely to change, to refer to e-mails and thus do not offer a stable identification. We show that MIDs are viable to identify and retrieve e-mails from an IMAP store in real-world scenarios. The presented concept can be integrated into any store, but we also offer a software solution that serves as an additional layer above the store and allows real-time access over MID. We propose a validation method to prove that the concept is working and some applications that are enabled by e-mail identification are sketched.

1 Introduction

Sending e-mails has long replaced traditional letters, especially in the business and research context. Being digital, e-mails can be easily stored and accessed from different locations, building up large archives as part of the personal information managed by each user. Although a central element of communication, e-mails are still tied to special software, the Mail User Agent, instead of being integrated into the overall workflow. Part of the reason is the lack of means to reliably identify an e-mail message within an archive. While the Internet Message Format–the de-facto e-mail standard–provides a globally unique Message-ID (MID) and this identifier is present in each e-mail, it is not used for identification [11]. Instead, e-mails are referred by their folder and a running number (UID)–both most likely to change if the user decides to put the message elsewhere or other messages appear [8].

Using a stable and persistent identifier for e-mails enables novel applications: Most archives are only accessible in a read-only mode, either due to technical reasons, e.g. WORM and similar media, or because of law regulations or policies. Therefore references cannot be adapted and identifiers need to be stable in the first place. Semantic applications store outside references to e-mails. With today's stores, those references become stale if a message is moved. As a result, references to e-mails are either not available or must be enforced by a very tight integration, locking out other applications. A further benefit of stable identifiers is the intrinsic cross-referencing given through the fact that identical messages have the same identifier, even across archives, i.e. an e-mail has the same MID at the sender and the receiver. When researching e-mail archives, for example in legal cases or historic research, this effect leads to higher efficiency.

We present a concept that enables referencing of e-mails by their Message-IDs. While this concept can be implemented into any software solution, we developed a prototype that works on top of IMAP-enabled stores, thus allowing additional functionality without changing running systems.

In the following, Related Work towards the topic is discussed. We then present the Message-ID Index, that adds MID-based references to an IMAP store, followed by a discussion of a validation method. Based on the availability of persistent identification, a set of enabled applications is sketched. The paper closes with a conclusion and the list of references.

2 Related Work

In [4] we argued towards persistent entity identification in Personal Information Management: It is necessary for recognition, dissemination and (external) cross-references to digital objects. *Uniform Resource Identifiers* (URIs) provide an established scheme for identification in the context of Internet communication and semantic technology [1].

Tools for alternative access to an IMAP store already exist. For example, [3] presents an IMAP plugin for *SquirrelRDF*[1] that allows to pose *SPARQL*[2] queries to an IMAP store. However, the proposed solution uses anonymous or generated node names for identifiers. While a *MessageID* attribute is provided, the paper clearly states that it is derived from the message number, which is even more volatile than the UID value and therefore should not be used.

The *Internet Message Format*, base for the e-mail format, is defined in RFC 5322 [11]. However, identification has been specified in a much earlier draft as RFC 724 [10]. RFC 2111 [6] specified a URI form of Message-IDs and is the base for the representation we chose to identify e-mails. The relevant parts of these specifications will be further elaborated in the next section. As the implementation resembles an offline IMAP store, many of the operations are described in RFC 4549 [7].

In [9] an analysis of stability and reliability of digital identifiers in *digital forensics*, including a survey on Message-IDs, is presented.

To the authors' knowledge current standard-conforming IMAP server implementations do not support an efficient query by Message-ID. The only known solution that uses an e-mail's Message-ID for identification is the *Gnowsis*[3] semantic desktop [12].

[1] http://notes.3kbo.com/squirrelrdf
[2] SPARQL is an acronym for "SPARQL Protocol And RDF Query Language". See http://www.w3.org/TR/rdf-sparql-query/ for further reading.
[3] http://www.semantic-web.at/de/gnowsis

3 Message-ID index

E-Mail Identification. Even without semantic applications and archiving, message identification is necessary for communication between a Mail User Agent (MUA) and an e-mail store. In the following we concentrate on the *Internet Message Access Protocol (IMAP)* as defined in RFC 3501 [2], which is supported by most server implementations and has become a wide-spread method for accessing e-mails from remote or distributed devices, e.g. from a desktop PC or smart phone. The IMAP standard defines the *Unique Identifier (UID) message attribute* as means of identification of a single message within the store. The UID is defined as an integer value that is unique within a specific IMAP folder. While this values is mean to be stable, the IMAP server may decide to re-organize the folder and thus change the UID for each message. Using folder and UID allows efficient storage and access during IMAP sessions, but is not viable for long-term identification.

A solution, however, is already embedded in each e-mail: The *Internet Message Format* defined in RFC 5322 [11], which is the base description for e-mails, contains a set of *Identification Fields* and, more specific, the `Message-ID` (MID). The MID is intended as a globally unique identifier embedded in each e-mail, which is currently used to generate threaded folder view, i.e. show the tree structure of messages within a single folder. There is a major downside to the Message-ID: The generation is left to the Mail User Agent. A malicious user can try to spoof an existing MID and mask other e-mails, if the MID is known. This attack is similar to other attacks on the message meta-data and providing a solution must be left to research on e-mail security. The form is roughly described in RFC 5322 as *localpart@domain*, were the domain should match the mail server's domain. The local part can either be a sequence number, a pseudo-random number or a hash of e-mail meta-data. Often a mixture is used in combination with the recipient address. However, efforts to create a recommendation for the Message-ID format never made it to the RFC catalog.[4]

Default operations of an IMAP store require fast access to folders and random access to messages within a folder, leading to a default hierarchy of message lists within a folder tree and the above identification scheme. While the MID is accessible from each single message, it does not help IMAP operations to make them available on a higher level, therefore this operation is not supported. There are IMAP implementations that use the Message-ID as file name for disk storage, but even then the folder tree, represented by directories, and the message lists apply.

To make the Message-ID quickly accessible, we propose an index on top of the IMAP structure that maps and updates Message-IDs to locations within the IMAP store. For an efficient solution in terms of runtime-efficiency, the index should be included in an IMAP server implementation. However, this would limit the use to a specific system and requires effort beyond the proof of concept, i.e. requires to adapt a software component that needs to be very reliable and can

[4] A draft can be found at http://tools.ietf.org/html/draft-ietf-usefor-message-id-01

cause severe data loss on misbehavior. It is, however, feasible to integrate the index into future implementations or create a specialized version that leverages features of a specific IMAP store.

Index Structure. The index is structured as follows:

$$\begin{aligned}
\langle\text{Message-ID}\rangle &\longmapsto (\langle\text{Folder URI}\rangle, \langle\text{UID}\rangle)_{\text{unique}} \\
\langle\text{Message-ID}\rangle &\longmapsto (\langle\text{reference type}\rangle, \langle\text{Message-ID}\rangle) \\
&\quad\ \text{with} \quad \langle\text{reference type}\rangle \in (\texttt{Reference}, \texttt{In-Reply-To}) \\
&\quad\ \text{and} \quad \texttt{In-Reply-To} \rightarrow \texttt{Reference}
\end{aligned}$$

Message-IDs are mapped to locations, which consist of a folder and the message's UID. The location pair is unique, i.e. there can be only one message at a specific location. However, since copying messages is allowed, the same e-mail can be found at several locations. While the resolution from a Message-ID to a message is unambiguous, the mapping to a location is not. The first line is sufficient to resolve MIDs, but we decided to index reference fields as well to allow quick searches for related e-mails. Those references are `In-Reply-To` for answers to a specific e-mail and `Reference` for a more generic reference, e.g. all e-mails from a discussion thread. The `In-Reply-To` field implies the `Reference` field and the reference is stored only once. Note that referenced e-mails are not necessarily available. For example, the user may have deleted the original e-mail before getting an answer, hence the `In-Reply-To` field points to an e-mail that does not exist locally. Still the identifier is valid and might be resolvable by a third-party user, e.g. the sender who kept the e-mail he answered.

Challenges. The main challenges are to create and update the index. Creation requires to crawl all available messages in order to extract their MID and references and add them to the index. While this process takes a while, it is only necessary on setting up the index. Subsequent runs can ensure consistency to avoid missed updates, but are by design not necessary. Keeping the index up to date is the larger problem. As the user moves messages around, a number of location entries change: First the moved message now is at a different location and the index entry for this message must be updated. The UID value should not change often, but if the server decides to re-organize the folder structure[5], all locations for this folder are invalid and must be retrieved again. For a responsive system, this re-indexing must be finished before the next Message-ID resolution.

The index implementation is very defensive against invalid entries. Therefore each location mapping is checked before it is returned to the caller. Checking is done by loading the message at the denoted location and comparing its Message-ID with the stored value. If there is no match, the entry is discarded as invalid. When no valid entries can be found, there is either no e-mail corresponding to the Message-ID in the store or the message is in a folder pending for re-indexing, in which case the resolution is stalled until the index is consistent

[5] The UIDVALIDITY attribute allows to detect when UID values have become invalid.

again. Regarding the latter case, there are two look-up contexts with different performance behavior: For resolution of a Message-ID to an e-mail, only one valid entry is needed, as all entries will point to the same content. However, for a list of all locations, the index must be consistent, otherwise the result may be incomplete. In a responsive user interface, this distinction should be made to avoid unnecessary long processing times.

Optimization. Having thorough checks on all mappings allows an optimization in the update process: When an e-mail is moved only the new location is stored. Since the same e-mail can be stored at different locations, a complete re-check of all locations would be necessary when a message is updated. This way, only the new entry must be added, which is a much faster operation. The other way around, uniqueness of locations allows to overwrite entries if the location mapping shows to a different Message-ID. These optimizations lead to faster indexing, but may result in an inconsistent index. Further performance tests are needed to decide whether these optimizations are necessary or if there is a faster solution.

Operations. Besides updates and consistency checks, which run automatically in the background, the index offers two main operations to a user: First, a Message-ID can be resolved either to its content, i.e. the complete message, or to the set of locations where the respective e-mail can be found. The main difference between those resolutions is that finding an e-mail requires only one valid location while finding all locations needs a completely consistent index and therefore has to wait until all update operations are finished. However, as the main purpose of the index is the resolution of Message-IDs to e-mail content, the faster query method will most likely be used. Second, the index allows to retrieve references to a specific e-mail, allowing to query preceding e-mails or related e-mails from an ongoing conversation. Note that this information come solely from the header fields within the message and are not inferred by further content analysis. If a Mail User Agent fails to set the respective fields, these references will not be available. However, apart from some Web-based mail systems, the data quality seems to be very good.

Architecture. Figure 1 shows the component diagram of the index. The Application has access to the underlying IMAP store for message content access and IMAP-related operations. Thus the default IMAP behavior is not hindered. To look up a Message-ID, the Application sends a Query to the Resolver, which in turn reads the mapping from an SQL DBMS. The result is checked against the IMAP store and, if valid, returned to the Application, otherwise invalidated in the DBMS. When the Resolver invalidates an entry, the Crawler is triggered, which in turn will read e-mails from a provided folder and update the respective mappings in the DBMS. The Observer is notified by the IMAP store if the message count in a folder changes, i.e. if messages are added, moved or removed, and triggers the Crawler to update the mappings. These notification can also

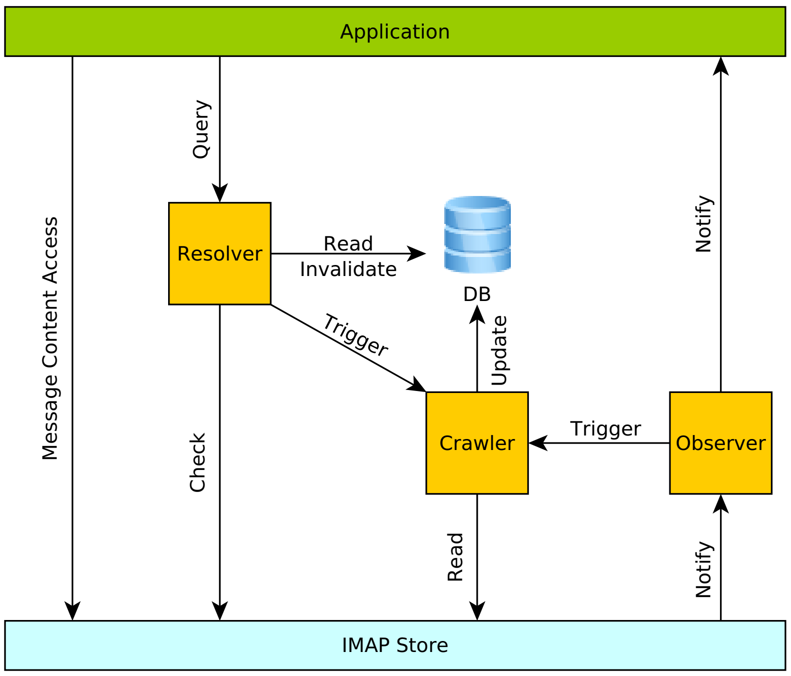

Fig. 1. Component diagram of the Message-ID index.

be received by the Application to update displays or trigger other application-dependent reactions.

Runtime Environment. We implemented the index using *Java*[6] and the *Java Mail API*[7] for generic IMAP access. For the IMAP store we are currently using the *Courier MTA*[8]. The index is stored in a *MySQL*[9] DBMS.

Our deployment environment is quite distributed, i.e. IMAP store, DBMS, index and application run on different machines in different networks. The first implementation could crawl 100.000 e-mails in about 2 hours. None of the machines showed substantial load, so most of the time goes into network communication. The current implementation uses parallelized crawling and bulk access to the database, so that with a rate of 50 e-mails per second the complete store can be crawled in half an hour.

[6] https://www.java.com/, the implementation uses the J2SE6 standard
[7] https://javamail.java.net/, Version 1.5.2
[8] http://www.courier-mta.org/imap/
[9] http://www.mysql.com/

4 Validation Methods

Due to time constraints, validation is not yet finished. However, we want to present our validation concept and first results.

The goal of validation is to show that with the index, resolution of Message-IDs to e-mail locations within an IMAP store is 1) faster than without an index and 2) fast enough for user interaction. Additionally, we measure the "usefulness" of the index at the time of access, i.e. the hit/miss ratio.

We expect that any of the supported operations is faster with the index than without. This follows from the rationale that a completely inconsistent or empty index would crawl the IMAP store, which is about the same operation as Message-ID lookup without an index. It is more interesting to see if the index is fast enough for user interaction. Studies have shown that a user becomes impatient after waiting two seconds for a response and annoyed after four seconds [13]. Since the look-up results from the index will most probably need further processing, even two seconds may be too much. However, we chose them as an upper limit: In an optimal operation, no query will take longer than two seconds. The *percentage of queries that take longer than N seconds* measures the performance, where $N = 2s$ can measure usability-critical performance. A better boundary for access times might be found from upcoming applications. First results show access times around 10 ms for a consistent index, which is coherent with the fact that only single SQL query and the retrieval of one e-mail header in an already open IMAP connection are necessary. For an inconsistent index, the access time is directly related to the time it takes to crawl the folder containing the message. Our current test system[10] achieves 50 e-mails per second. The folder structure in this system consists of smaller "work" folders with less than 100 e-mails and large "archive" folders with up to 15.000 e-mails. Crawling the larger folders takes several minutes, but only occurs when the UIDVALIDITY changes, which has not happened in 3 months of running the system. The smaller folders undergo much more fluctuation, but can be scanned within 2 seconds. The observation component even shortens these values: Changes to a single message are reported within one second with a rate of 20 messages per second for bulk operations.[11] For most of the time, the 2-seconds-limit for usable access was met.

The *readiness* of the index can by evaluated by the ratio between hits, i.e. found entries, and misses in terms of invalid entries or entries that where not available due to re-indexing. It is important not to evaluate if entries could not be found, as they may come from Message-IDs for e-mails that were never available in the store. To count as a *miss* the entry should have been available in a perfect index. We expect misses on two occasions: Either a message has been added, but is not yet available in the index or messages have been removed

[10] A quad-core Intel Atom platform. However, these values are only first estimates as the test setup is not yet fit for clean statistics.

[11] IMAP bulk operations appear when a large number of messages are marked as read, as supported folder-wise by many clients, or a large number of messages is moved to another folder.

from a folder, but re-indexing is still pending. The observer module is directly informed about e-mails that have been added to a folder, so new messages are available very quickly. However, it may take time to re-organize if the message numbers have changed. So in most cases not the moved, but the remaining messages are affected. We expect that in a real scenario this lessens the number of misses, as the focused message is always readily available. However, re-indexing is relevant if an application frequently requests all storage locations. Furthermore we measure the *time the index spends in an inconsistent state* with pending re-indexing. We observed only two scenarios where the index was not ready within one second after a change: 1) When emptying the trash, as each message is reported individually. 2) When marking a large number of messages read, which happens often for mailing list conversations. As observation and crawling are parallelized processes, other folders were not affected and changes in these folders were available within one second. Therefore we could not observe a significant amount of cache misses.

A trivial approach towards a test scenario is a random distribution of operations (add, move, delete e-mails) and accessed e-mails. However, this pattern is far away from typical use cases. After their receipt only a small fraction of e-mails will ever be accessed again. As a consequence, there is a rather small set of "hot" e-mails a user might ask for, while the larger part will never be accessed. We suspect that those relatively new e-mails are in a distinct set of folders, like the INBOX or folders based on current project, while the rest has been moved to archive folders where they will most likely stay, so that for every (re)move operation only a very small part of messages is affected. Therefore, a more realistic test-scenario will weight the access based on the age of an e-mail and on the folder. A third test scenario comes from user observation. Due to privacy reasons it is however very hard to get the respective data. Observing user behavior on e-mails relies on an index on the user's e-mail store and tracking of live usage data in a critical part of everyday communication. Therefore the test set will only be very small. While the last scenario is hard to acquire, we expect the best evaluation results since it matched actual access patterns best.

5 Outlook on Applications

In this section we present three applications that are enabled by fast Message-ID resolution.

Related E-Mails. The first application allows to view related messages in a *Mail User Agent (MUA).* While a tree view within a folder is often available, it is not possible to display a series of messages across folder boundaries, since related messages cannot be found fast enough. With the index, however, related messages can be easily retrieved and displayed in a tree alongside a selected message. This allows the user to easily navigate through a set of messages, regardless of the folder they are stored in. Since older messages are often moved to archive folders, this application allows the user to see all related messages even if they have already been archived.

Misplaced E-Mails. Related e-mails are often stored in the same folder, e.g. based on a project or the communication partner. When a message is accidentally misplaced it is very hard to recover this message later on.[12] A quick access to message references, provided by the second part of the index, allows to check if there are any stray messages. While it cannot be assumed how a user organizes e-mails, the amount of e-mails references that cross folder boundaries can be used to determine if a message may be in the wrong folder and where it should be. An appropriate notification can inform the user about the potential mishap and offer a quick solution.

External References. A third application directly uses the fact that Message-ID references can be easily resolved: Using an add-on, the browser can be enabled to understand the MID URI scheme. When confronted with a respective URI, the message will be looked up and displayed in the browser or opened directly in the MUA. As a result it is now possible to send links to e-mail messages, e.g. via e-mail, instant messenger or embedded into a document, regardless where they are stored.

6 Conclusion

Although e-mails play an important role in modern communication, there is no applicable way of referencing them. We have shown that a generic IMAP store can be enhanced so that the already existing Message-ID field is viable for persistent, long-term identification and reference of e-mail messages in archives and semantic applications. We have also presented a set of applications that are enabled by fast e-mail identification via Message-ID.

The validation concept is ready and our next step will be a long-term validation of the index on realistic scenarios to proof its usefulness. As mentioned before, evaluation on real data is challenged by privacy issues. During further research we will try to build a test set based on the ENRON e-mail dataset [5] to diminish the issue of private test data. We were also able to find a number of people who are willing to run an analysis tool on their mailbox to acquire accumulated statistics about folder structures and message distribution over time.

For the software itself, further implementation will include more statistics for the validation process and a better recoverability on link failures to the IMAP server.

Afterwards the applications sketched in the outlook will be implemented.

References

1. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (January 2005), http://tools.ietf.org/html/rfc3986
2. Crispin, M.: Internet Message Access Protocol - version 4rev1. RFC 3501 (March 2003), http://tools.ietf.org/html/rfc3501

[12] The best way to hide a book in a library is to put it into another shelf.

3. Eynard, D., Recker, J., Sayers, C.: An IMAP plugin for SquirrelRDF. Tech. rep. (October 2007), http://www.hpl.hp.com/techreports/2007/HPL-2007-161.html
4. Haun, S., Nürnberger, A.: Towards persistent identification of resources in personal information management. In: Predoiu, L., Mitschick, A., Nürnberger, A., Risse, T., Ross, S. (eds.) SDA. CEUR Workshop Proceedings, vol. 1091, pp. 73–80. CEUR-WS.org (2013), http://dblp.uni-trier.de/db/conf/ercimdl/sda2013.html#HaunN13
5. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS (2004), http://dblp.uni-trier.de/db/conf/ceas/ceas2004.html#KlimtY04
6. Levinson, E.: Content-ID and Message-ID Uniform Resource Locators. RFC 2111 (March 1997), http://tools.ietf.org/html/rfc2111
7. Melnikov, A.: Synchronization Operations for Disconnected IMAP4 Clients. RFC 4549 (June 2006), http://tools.ietf.org/html/rfc4549
8. Melnikov, A., Newman, C.: IMAP URL Scheme. RFC 5092 (November 2007), http://tools.ietf.org/html/rfc5092
9. Pasupatheeswaran, S.: Email 'Message-IDs' helpful for forensic analysis? In: Proceedings of the 6th Australian Digital Forensics Conference. School of Computer and Information Science, Edith Cowan University, Perth, Western Australia (2008)
10. Pogran, K., Vittal, J., Crocker, D., Henderson, A.: Proposed Official Standard for the Format of ARPA Network Messages. RFC 724 (May 1977), http://tools.ietf.org/html/rfc724
11. Resnick, P.: Internet Message Format. RFC 5322 (October 2008), http://tools.ietf.org/html/rfc5322
12. Sauermann, L.: The Gnowsis Semantic Desktop for Information Integration. In: Proceedings of the IOA 2005 Workshop at the WM. Springer (2005), http://www.dfki.uni-kl.de/s̃auermann/papers/Sauermann2005a.pdf
13. Shneiderman, B.: Designing the user interface: strategies for effective human-computer interaction. Addison-Wesley, Reading (1998)