The KDE Kroupware Client

Bo Thorsen

Table of Contents

Free Software in the BSI

The Kroupware project is a development that Bundesamt für Sicherheit in der Informationstechnik (BSI) of the German government contracted from a consortium of three companies - Intevation, Erfrakon and Klarälvdalens Datakonsult AB.

The project has three parts:

- A groupware server.
- A KDE client.
- Compatibility with Microsoft Outlook.

The server has been made by Erfrakon; the client by Klarälvdalens Datakonsult AB and Intevation handled project administration. Outlook compatibility is a shared task between the parts.

BSI wanted to do a move towards free software, both on the client and on the server. Currently there are no solutions that provide this completely, although several projects are close.

The solution we presented them is not a new development per se, but instead an integration of existing parts and projects. Building a complete groupware solution is an incredibly large job, but by using the existing parts, we were able to have a working prototype in less than three weeks.

The Kolab Server

The server built by Erfrakon is a collection of free software server parts that have been put together to form a complete groupware server solution. On top of the parts, Erfrakon built a web administration module.

The server parts are:

- Apache Web server
- Postfix Mail server
- Cyrus Imap server
- OpenLDAP LDAP server
- Proftpd FTP server
- Sieve scripting

A couple of other parts were also used, but these are the main ones.

The webinterface they built is used to administrate these parts in an integrated way. For example, adding a user will put the user information in the addressbook supplied by the LDAP server, it will make a directory for web access using webdavs (and ftp, if legacy support is enabled), and so on. Screenshots of the user administration is given below.

The webinterface also controls the entire server environment. It is here one can en- or disable services, as shown in the next screenshot.

As you can see in the screenshot, ftp is only used for Outlook 2000 compatibility. Using this old and insecure protocol is an unfortunate necessity when maintainting compatibility with old legacy software. There are also many places in the client where this compatibility requirement have meant lots of sidestepping the normal Outlook approximations to standards.

To make Outlook work with the server, we are using the Bynari plugin, that enables Outlook to use other servers than Microsoft Exchange. This plugin is also used in other Kolab like servers. Currently SuSE and Bynari provides commercial Exchange replacements that provide functionality akin to Kolab. But Kolab is the only free software server that does the job.

I will not describe the server in more detail here, but instead go over the parts as seen from the client view in the discussion of this.

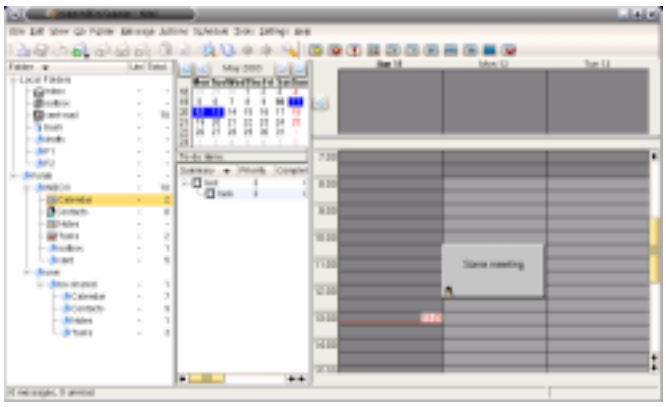
The KDE Kroupware Client

Where the server parts were easy to determine (setting aside discussions like which IMAP server should be used), the client was potentially taken from a larger array of possibilities. Mozilla and Evolution would be the two other solutions that would have been fairly easy to get working with Kolab. Yet another possibility would be to extend an existing webbased groupware solution, but none was found to be stable enough.

A KDE solution based on KMail was the one that was chosen. Among the arguments for this was the Ägypten project that Intevation, Klarälvdalens Datakonsult AB and g10 Code implemented for BSI.

It was decided that the main userinterface would be to have the KDE calendar application KOrganizer embedded in KMail. Additionally, the KDE addressbook KAddressbook should be able to access the LDAP addressbook in the Kolab server. We did not want to rename the actual parts used, so by now we use KMail, KMKO (abbreviation of KMail/KOrganizer) or the Kolab client as the term for the application.

The next screenshot shows the main user interface of the client.



New KMail features

Klarälvdalens Datakonsult AB and Marc Mutz added a good list of new features to KMail for the Kroupware project:

- Disconnected IMAP
- Message Disposition Notification
- Automatic Resource Handling
- IMAP resource backend
- KOrganizer connectivity

In the next sections, I will describe each of these.

The Disconnected IMAP account

Disconnected IMAP - from here on dIMAP - is at the heart of the Kroupware project, since all user info is stored in IMAP folders on the server.

dIMAP works by synchronizing the IMAP server contents to the local disk. The KMail user will after the sync work on the local cache, and upon the next synchronization, all local changes will be uploaded to the server before new mails are downloaded to the local cache. The basic idea is that after the synchronization, the contents of the server and the local cache are identical.

The user benefits in this is that you do not have to be online to go through the mail boxes. It is also an easy way to make two machines have the same mail setup - for example your workstation and a laptop.

With the current IMAP implementation in KMail no local cache is used which means the user must be online while going through the mail boxes. The biggest user benefit in this is that you local disk usage is almost zero - not a small benefit if you have several gigabytes of email.

The future work for dIMAP includes two enhancements: Selecting a folders to be synchronized, and creating filters for what will be synchronized.

Folder selection have almost been implemented. The idea here is that normally you won't synchronize all folders - users often have old mail archives that are not often changed.

Synchronization filters is a setup where the user can select parts of the folders that shouldn't be downloaded to the local cache (unless explicitly told to do so). Examples include "Don't cache mails larger than X kb", "Don't cache attachments" and "Don't cache mails older than two months".

Comparing the two IMAP implementations show that the current IMAP implementation is almost the same as the behaviour you get when using a dIMAP implementation with a sync filter set to "Don't cache anything" and the user deleting the cache after reading a mail. The current implementation is very handy in the situation where users have very little disk space available. When this isn't the case, the dIMAP implementation should prove more userfriendly.

dIMAP have one other big advantage over normal IMAP. With this, it's possible to use KMails excellent mail filtering.

The implementation of dIMAP is approaching a very stable situation, but the HEAD implementation still experiences crashes. Data loss haven't been seen in testing since november, but the local cache has been seen to be confused. Advice to people wanting to use dIMAP is to backup mails they really don't want to loose. The implementation is definately stable enough for more thorough testing so we can find the remaining bugs that are there. Speed-wise, the implementation isn't as effective as it could be. We have concentrated completely on stability and sacrificed all speedups we could think of to make the implementation as stable as it can be. The

effect should be that it’s much more stable, but synchronization can take quite a long time. Later work on this should yield big speedups in the synchronization.

dIMAP uses the normal KMail maildir for the local cache, so technically, what we did was to make the synchronization code and not touch the local storage. This means that the local cache is just as stable as having normal local maildir folders. At a later point we hope to decouple the account code completely from the folder code, so the user can choose which type of folder he wants as the local cache.

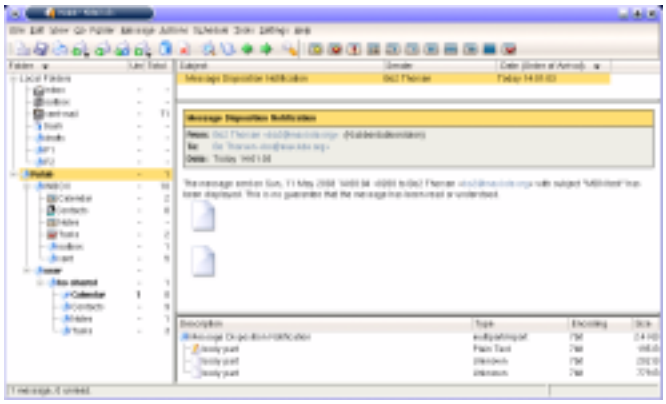
In all KMail screenshots that are here, the folders in the Kolab account are dIMAP folders. The folders under INBOX are the private user folders. The folders under shared are shared between groups of persons - shared folders are set up by the web admin module

Message Disposition Notification

MDNs - Message Disposition Notifications - are a generalization of what is commonly called "read receipt". The message author requests a disposition notification to be sent and the receiver’s mail program generates a reply from which the author can learn what happened to his message. Common disposition types include "displayed" (i.e. read), "deleted" and "dispatched" (e.g. forwarded).

In KMail you can request notification when people read the mail you sent, and KMail can send notification when you read mail sent to you with notification requests.

The next screenshots show the dialog you get when opening an MDN mail, and the returned mail.



Vacation setup

You can now setup timeframes where you are away from a small userfriendly dialog in KMail:

Automatic resource scheduling

In groupware mode, you can setup accounts to work as automatic resource scheduling. With this, you set up accounts for each resource you want to schedule. An example could be a room. You make an account that holds the rooms calendar, and people "invite" the room to a meeting. When KMail receives such an invitation, it automatically accepts if the resource is free - if not, it rejects the invitation. We don't mind people calling this a hack, but it works surprisingly well :-)

IMAP Resource backend to KMail

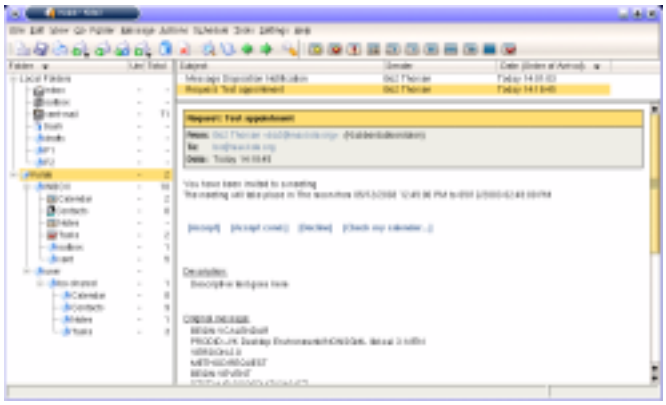
The new resource framework in kdelibs and kdepim have different backend implementations. The obvious one is to use the local disk for the storage, which is the standard implementation. For the addressbook, there is also an LDAP backend, and so on.

We have implemented a resource backend in KMail, so this can be used for storage. This means KOrganizer, KAddressbook and other applications can save their data in KMail folders. This is specifically done with dIMAP in mind, so synchronization in addition to emails also syncs calendar, contacts and notes

KOrganizer connectivity for groupware scheduling

To complete the groupware functionality, code have been added to KMail to intercept incoming groupware messages.

In KMail, different things happen depending on what is in the message currently being read. We have added the check for mimetypes holding calendar related information. When a message with such a mimetype is seen, the iCal file is given to the groupware code that reformats it to a HTML message which includes the choices the user has. So when an invitation arrives, this is transformed to a message saying you have been invited to a meeting in XX from YY to ZZ. Below that there are four links with different options, the most important being accept and decline. When the user clicks one of these links, KMail will call KOrganizer code with the message and the user choice. After that it is up to KOrganizer to put the information into the right resource (which could be the IMAP resource as described above). A screenshot of receiving an invitation is shown below.



In addition to this, KMail also offers to format and send the messages from KOrganizer, so correct replies and invitations are sent. More on this below.

The general split here is that KOrganizer handles all iCal related, and KMail handles how the iCal is put into mails or read from mails.

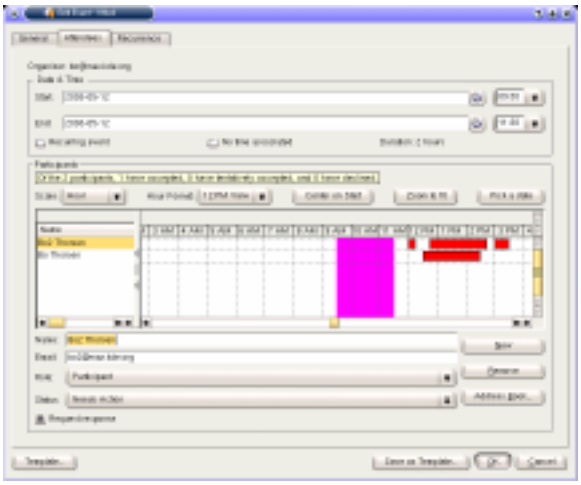
KOrganizer enhancements

Klarälvdalens Datakonsult AB also did the work on KOrganizer and the underlying iCalendar libraries to make it work with the Kolab server inside KMail.

There were basically three things we had to do:

- Enhance groupware scheduling
- Enable IMAP (KMail) storage
- Embed KOrganizer in KMail

KOrganizer already had a kind of groupware scheduling, but it was missing several features. The most visible of these was fixed with the addition of a gantt view in the invitation dialog. A screenshot of this is given below.



In this screenshot, you can also see the list of people invited, you have the action that these are expected to take, their roles, and overview of who have answered what and so on. The gantt view shows the schedule of each of the participants, with the red boxes being the busy periods, and the purple vertical bar being the current time of the meeting. The dialog can automatically find the next time where all participants are free.

KOrganizer is embedded in KMail using the KParts technology. This is the component architecture in KDE that enables parts containers to embed parts inside the user interface. Especially Konqueror uses this extensively to show different file types.

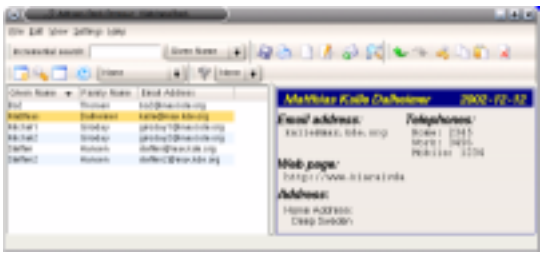
Not directly visible was all the work that was done on KOrganizer and the kdepim libraries to make it work with KMail. This work included answering KMail requests about iCalendar files; constructing iTIP messages (iCalendar scheduling files) to be sent by KMail; receiving appointment replies and updates; and using KMail IMAP folders as the storage.

Other Kdepim Components

Two more applications needed some extra work - KPilot and KAddressbook. Unlike KOrganizer, these continue to be separate applications as opposed to run inside KMail.

KPilot is the application that syncs calendar and addressbook information to Palm Pilot handhelds. This needed to be modified so it was also able to read everything from the KMail storage.

KAddressbook also got that extra feature. In the Kolab environment this has a special meaning. The shared addressbook is shared using LDAP. The entries in the users own Contacts KMail folder are private entries, where the user can add all contact information. A screenshot of KAddressbook with some local contacts is shown below.



Future Work

In kroupware_branch all this is working, but in cvs HEAD there are still some issues to be sorted out before this is working. The difference is that in kroupware_branch, all communications is done with Qt signals and slots, where in HEAD it's done with DCOP. This is currently being retrofitted to work in the Kontact framework and is looking very promising.

Kontact is better designed technically, and will hopefully also be an even better user experience. Currently the KMail parts of kroupware have been ported, and the IMAP resource backend have been reimplemented in the new framework. The missing parts are the korganizer parts that enable the groupware scheduling. This will hopefully be done before KDE 3.2.

A screenshot taken by Zack Rusin of Kontact:

Other future work is to implement a lot more groupware features. The Kroupware project had a very specific list of features, and there are plenty more features that we would like to implement.

Another area is that we hope other projects will implement support for the Kolab server and cooperation with the KDE client. A webbased solution is currently in it's early beginning, and hopefully Evolution and Mozilla people will work on this too.

Reference Links

The three companies that implemented the Kroupware Project:

- Klarälvdalens Datakonsult AB [http://www.klaralvdalens-datakonsult.se]
- Intevation [http://www.intevation.net]
- Erfrakon [http://www.erfrakon.de]

The project and the KDE parts

- The Kroupware Project [http://www.kroupware.org]
- KMail [http://kmail.kde.org]
- KDE PIM applications [http://kdepim.kde.org]

- KMail [http://kmail.kde.org]
- Kontact [http://kontact.kde.org]

Other projects

- The Ägypten project [http://www.gnupg.org/aegypten/]