Administration Manual for UrBackup

Martin Raiber

August 9, 2011

Contents

1 Introduction

UrBackup is a client/server backup system. This means there exists a server which backups clients. Accordingly UrBackup is divided into a client and server software. The client software currently runs only on Windows while the server software runs on both Linux and Windows.
UrBackup is able to backup files and images. The clients have to define the paths where the files to be backed up are. Images are automatically taken of the system drive (C:).
A lot of effort in UrBackup was made to make setup as easy as possible. If you are happy with the default settings (see section 6) the only thing you need to define on the server side is where backups should be stored. On the clients you only need to say which directories should be backed up. If server and clients are in the same subnet the server will automatically discover the clients and then start backing them up (for details see section 4). This also makes building a decentralized backup strategy very easy, as e.g. one backup server per subnet is responsible for backing up all clients in this subnet. If a computer is moved from one subnet to another this new client is discovered and the server in the new subnet automatically takes over backing it up. If you want to implement something like this you should also read the section on security (see 3) for details on when a client accepts a server.
The interested administrator should read up on the general UrBackup architecture (section 2), how the backups are stored and performed (section 5), proposals on which filesystems are suited (section 7.3) and take a look at some sample a setup with ZFS at section 7.4.1.

2 Architecture

As already mentioned UrBackup is divided into a server and a client software. The server is responsible for discovering clients, backing them up, deleting backups if the storage is depleted or too many backups are present, generating statistics and managing client settings. The client is relatively dump. It listens to server commands which tell it e.g. that a file list should be build or which file the server wants to download. The server also starts a channel on which the clients can request the server to start a backup or to update the client specific settings.

2.1 Server architecture

The server is organized into a core part and an interface. Currently only a webinterface is available. The webinterface is accessible via FastCGI (on port 55413) and HTTP (on port 55414). You can use the FastCGI port to make the webinterface accessible via SSL (using e.g. apache webserver). For details on that see section 3.2. The server core part consists of several threads with different tasks. One thread discovers new clients, another checks if a client needs to be backed up, while others send pings to clients to see if they are still alive or send them the current backup status. One updates file statistics or deletes old backups. Because there are so many threads UrBackup server profits from modern multi core CPUs (the more cores the better!).

2.2 Client architecture

The client is divided into a core process and an interface process. The interface process displays the tray icon and the dialogs and sends settings and commands to the core client process. The core client process listens on port 35622 UDP for UDP broadcast messages from the server and on receiving one sends a message with its name back to the server. As name the Windows computer name is used. It listens on port 35623 TCP for commands from the client interface process and the server and on port 35621 TCP for file requests from the server. The server establishes a permanent connection to each client on its command port with which the clients can request backups or change their settings. The core client process is responsible for building a list of all files in the directories to be backed up. This list is created in the UrBackup client directory as 'urbackup/ data/ filelist.ub'. To speed up the directory list creation directories to be backed up are constantly watched via the Windows Change Journal. The Windows Change Journal can only be used for whole partitions. Thus the first time a directory on a volume is added the UrBackup core client process reads all the directory entries on the new volume into the client database file in 'urbackup/backup_client.db'. After a volume is successfully indexed the database is constantly updated to be in sync with the filesystem. Thus if large changes in the volume occur the database gets updated more often. This does not have a big performance penalty as only directories are saved in the database. The updating is done every 10 seconds or if a file list is requested. The server downloads the file list from the client and starts the backup by downloading changed or new files from the build in client file server. The image backup is done using only the command port.

3 Security

3.1 Server webinterface rights management

The server webinterface is protected by a pretty standard user system. You can create, manage and delete accounts. Those accounts are only linked loosely to clients by rights management. Be aware that after first installing UrBackup there is no administrator password set and everybody can see all backuped files! If you want to limit access you should immediately go to the account management in the settings and create an administrator account and set its password.
An admin account can do everything including browsing file backups of all clients. The webinterface allows one to create a 'limited' account that can only browse backups and view statistics from one client. The more sophisticated rights editor can be used to allow an account to access several clients or to limit some aspects. For example you could setup an account which can do everything except browse backups. Following domains, with which you can limit or expand an accounts rights, are currently available:

Domain	Description
browse_backups	Browse and download files from file backups
lastacts	View the last actions (file or image backups) the server did (including backup size and duration)
progress	View the progress of currently running file or image backups
settings	Allows settings to be changed
status	Allows the current status to be viewed (last seen, last file backup and last image backup)
logs	View the logs which were creating during backups
piegraph*	View statistics
users*	Get client names
general_settings*	Change general settings (like backup storage path)
usermod*	Create, change and delete users
remove_client*	Remove clients and delete all their backups

You can set the domains not marked with stars(*) either to one or several client ids (separated by ',') or to 'all' - meaning the account can access all clients. The entries with stars(*) have to be set to 'all' or 'none' and don't allow client ids. In order to be able to view statistics you need to set both 'piegraph' and 'users' to 'all'. There is a special domain 'all' which is a wildcard for all domains (this means if you set 'all' to 'all' the account has the right to do everything).

3.2 Make webinterface accessible via SSL

The server webinterface is accessible via FastCGI (on port 55413 TCP). With this you can connect UrBackup with pretty much every modern web server and thus make the webinterface accessible via SSL. This section will describe how to do this with apache and lighttp.

3.2.1 Apache configuration

Either add a symlink to the 'www' UrBackup directory or define it as an alias. For the symlink method you need to go to your SSL webroot and then do e.g.:

```
ln -s /var/lib/urbackup/www urbackup
```

Be sure you have set 'Option +FollowSymLinks' in the webserver configuration on the directory you link into. From now on it is assumed that urbackup should be accessible via https://hostname/urbackup. Download and install 'libapache2-mod-fastcgi' (this may have another name on other distributions). Add following line to the 'fastcgi.conf':

```
FastCgiExternalServer /var/www/urbackup/x -host 127.0.0.1:55413
```

The path depends of cause on where your web root is and where you want the web interface to be. UrBackup should now be accessible via apache.

3.2.2 Lighttp configuration

Link the urbackup/www directory into the webroot as described in the apache configuration. Add

```
include "conf.d/fastcgi.conf"
```

to your 'lighttp.conf' file. Then add

```
fastcgi.server = (
  "/urbackup/x" =>
  (( "host" => "127.0.0.1",
     "port" => 55413
  ))
)
```

to the 'fastcgi.conf' file.

3.3 Client security

UrBackup Client only answers commands if the server or the interface process supply it with credentials. The server credential is saved in '/var/ lib/ urbackup/ server_ident.key'. If it does not exist the server will randomly generate it the first time it runs. The client interface credential is generated in the same way and resides in 'pw.txt' in the UrBackup directory on the client. To give the client core process interface commands you need the contents of 'pw.txt'. The client core process saves the server credentials from which it accepts commands and which it allows to download files in 'server_idents.txt' - one credential per line. You need to remove the preceding '#I' and '#' at the end of the contents of 'server_ident.key' if you want to add a server identity to 'server_idents.txt'. After installation the 'server_idents.txt' does not exist and the client core process accepts(and adds) the first server it sees. After that no other servers with different credentials are accepted and you need to add their credentials manually. This prevents others from accessing files you want to be backed up in public places.
If you want to have several servers to be able to do backups of a client you have two options. Either you manually supply the server credentials to the client (by copying them into 'server_idents.txt') or you give all servers the same credentials by copying the same 'server_ident.key' to all servers.

3.4 Transfer security

The transfer of data from client to server is unencrypted allowing eavesdropping attacks to recover contents of the data that is backed up. With this in mind you should use UrBackup only in trusted local networks. If you want to use UrBackup over an untrusted network like the Internet for security (and technical - see section 4) reasons use a VPN solution that provides end-to-end encryption (such as OpenVPN, IPSec, ...).

4 Client discovery

UrBackup clients should be discovered automatically given that server and client reside in the same subnetwork. The client discovery works as follows:
The UrBackup server broadcasts a UDP message every 50 seconds on all adapters into the local subnet of this adapter. On receiving such a broadcast message the client answers back with its name. Thus it may take up to 50 seconds until a client is recognized as online.
If the client you want to backup is not in the same subnet as the server you can add its IP or hostname manually by clicking "show details" in the settings and then adding an "extra client". The server will then additionally send an UDP message directly to that entered IP or resolved hostname allowing routers to forward the message across subnet boundaries. Be aware though that all connections are from server to client, e.g. if you use NAT you need to forward the client ports (35622 UDP, 35621 TCP, 35623 TCP) to the client. Currently there is no option to change these ports, so you would be limited to just one client if you have NAT. You should use VPN in this case.

5 Backup process

This section will show in detail how a backup is performed.

5.1 File backup

- The server detects that the time to the last incremental backup is larger then the interval for incremental backups or the last time to the last full backup is larger then the interval for full backups. Backups can be started on client requests as well.
- The server creates a new directory where it will save the backup. The schema for this directory is YYMMDD-HHMM with YY the year in a format with two decimals. MM the current month. DD the current day. And HHMM the current hour and minute. The directory is created in the backup storage location in a directory which name equals the client name.
- The server requests a file list construction from the client. The client constructs the file list and reports back that it is done.
- The server downloads 'urbackup/data/filelist.ub' from the client. If it is an incremental backup the server compares the new 'filelist.ub' with the last one from the client and calculates the differences.
- The server starts downloading files. If the backup is incremental only new and changed files are downloaded. If the backup is a full one all files are downloaded from the client.
- The server downloads the file into a temporary file, thus enough space on the temporary file location should be available. On successfully downloading a file the server calculates its hash and looks if there is another file with the same hash value. If such a file exists they are assumed to be the same and a hardlink to the other file is saved and the temporary file

deleted. If no such file exists the file is moved to the new backup location. File path and hash value are saved into the server database.

- If the backup is incremental and a file has not changed a hardlink to the file in the previous backup is created.
- If the client goes offline during the backup and the backup is incremental the server continues creating hardlinks to files in the previous backup but does not try to download files again. The files that could not be downloaded are then not saved into the server side file list. If the backup is a full one and the client goes offline the backup process is interrupted and the partial file list is saved, which includes all files downloaded up to this point.
- If all files were transfered the server updates the 'current' symbolic link in the client backup storage location to point to the new backup. This only happens if the client did not go offline during the backup.

5.2 Image backup

The server detects that the time to the last full image backup is larger then the interval for full image backups, the time to the last incremental backup is larger than the interval for incremental image backups or the client requested an image backup. The server then opens up a connection to the client command service requesting the image of a volume (for now only the C: volume). The client answers by sending a error code or by sending the image. The image is sent sector for sector with each sector preceded by its position on the hard disk. The client only sends sectors used by the filesystem. If the backup is incremental the client calculates a hash of 256 kbyte chunks and compares it to the previous image backup. If the hash of the chunk has not changed it does not transfer this chunk to the server, otherwise it does. The server writes the received data into a temporary file. The temporary files have a maximum size of 1GB. After this size is exceeded the server continues with a new temporary file. The image data is written to a VHD file in parallel in a separated Thread and is located in the client directory in the backup storage location. The VHD file's name is 'Image_<Volume>_<YYMMDD_HHMM>.vhd'.<Volume>being currently always 'C' and YY the current year, MM the current month, DD the current day in the month and HHMM the hour and minute the image backup was started.

6 Server settings

The UrBackup Server allows the administrator to change several settings. There are some global settings which only affect the server and some settings which affect the client and server. For those settings the administrator can set defaults or override the client's settings.

6.1 Global Server Settings

The global server settings affect only the server and can be changed by any user with "general_settings" rights.

6.1.1 Backup storage path

The backup storage path is where all backup data is saved. To function properly all of this directories' content must lie on the same file system (otherwise hardlinks cannot be created). How much space is available on this filesystem for UrBackup determines partly how many backups can be made and when UrBackup starts deleting old backups. Default: "".

6.1.2 Do not do image backups

If you check this the server will not do image backups at all. Default: Not checked.

6.1.3 Automatically shut down server

If you check this UrBackup will try to shut down the server if it has been idle for some time. This also causes too old backups to be deleted when UrBackup is started up instead of in a nightly job. In the Windows server version this works without additional work as the UrBackup server process runs as a SYSTEM user, which can shut down the machine. In Linux UrBackup server runs as a limited user which normally does not have the right to shut down the machine. UrBackup instead creates the file '/var/lib/urbackup/shutdown_now', which you can check for existence in a cron script e.g.:

```
if test -e /var/lib/urbackup/shutdown_now
then
shutdown -h +10
fi
```

Default: Not checked.

6.1.4 Autoupdate clients

If this is checked the server will automatically look for new UrBackup client versions. If there is a new version it will download it from the Internet and send it to its clients. The UrBackup client interface will ask the user to install the new client version. The installer is protected by a digital signature so malfeasance is not possible. Default: Checked.

6.1.5 Max number of simultaneous backups

This option limits the number of file and image backups the server will start simultaneously. You can de- or increase this number to balance server load. A large number of simultaneous backups will of course increase the time the server needs for one backup, if many backups are run in parallel. The number of possible simultaneous backups is virtually unlimited. Default: 10.

6.1.6 Max number of recently active clients

This option limits the number of clients the server accepts. An active client is a client the server has seen in the last two month. If you have multiple servers in a network you can use this option to balance their load and storage usage. Default: 100.

6.1.7 Cleanup time window

UrBackup will do its cleanup during this time. This is when old backups and clients are deleted. You can specifiy the weekday and the hour as intervals. The syntax is the same as for the backup window. Thus please see section 6.2.1 for details on how to specifiy such time windows. The default value is 1-7/3-4 which means that the cleanup will be started on each day (1-Monday - 7-Sunday) between 3 am and 4 am.

6.2 Client specific settings

Settings	Description	Default value
Interval for incremental file backups	The server will start incremental file backups in such intervals.	5h
Interval for full file backups	The server will start full file backups in such intervals.	30 days
Interval for incremental image backups	The server will start incremental image backups in such intervals.	7 days

Interval for full image backups	The server will start full image backups in such intervals.	30 days
Maximal number of incremental file backups	Maximal number of incremental file backups for this client. If the number of incremental file backups exceeds this number the server will start deleting old incremental file backups.	100
Minimal number of incremental file backups	Minimal number of incremental file backups for this client. If the server ran out of backup storage space the server can delete incremental file backups until this minimal number is reached. If deleting a backup would cause the number of incremental file backups to be lower than this number it aborts with an error message.	40
Maximal number of full file backups	Maximal number of full file backups for this client. If the number of full file backups exceeds this number the server will start deleting old full file backups.	10
Minimal number of full file backups	Minimal number of full file backups for this client. If the server ran out of backup storage space the server can delete full file backups until this minimal number is reached. If deleting a backup would cause the number of full file backups to be lower than this number it aborts with an error message.	2
Maximal number of incremental image backups	Maximal number of incremental image backups for this client. If the number of incremental image backups exceeds this number the server will start deleting old incremental image backups.	30
Minimal number of incremental image backups	Minimal number of incremental image backups for this client. If the server ran out of backup storage space the server can delete incremental image backups until this minimal number is reached. If deleting a backup would cause the number of incremental image backups to be lower than this number it aborts with an error message.	4
Maximal number of full image backups	Maximal number of full image backups for this client. If the number of full image backups exceeds this number the server will start deleting old full image backups.	5
Minimal number of full image backups	Minimal number of full image backups for this client. If the server ran out of backup storage space the server can delete full image backups until this minimal number is reached. If deleting a backup would cause the number of full image backups to be lower than this number it aborts with an error message.	2
Delay after system startup	The server will wait for this number of minutes after discovering a new client before starting any backup	0 min
Backup window	The server will only start backing up clients within this window. See section 6.2.1 for details.	1-7/0-24
Excluded files	Allows you to define which files should be excluded from backups. See section 6.2.2 for details	""
Default directories to backup	Default directories which are backed up. See section 6.2.3 for details	""
Allow client(s) to change settings	If this option is checked the clients can change their client specific settings via the client interface. If you do not check this the server settings always override the clients' settings.	true

6.2.1 Backup window

The server will only start backing up clients within the backup windows. The clients can always start backups on their own, even outside the backup windows. If a backup is started it runs till it is finished and does not stop if the backup process does not complete within the backup window. A few examples for the backup window:
1-7/0-24: Allow backups on every day of the week on every hour.
Mon-Sun/0-24: An equivalent notation of the above
Mon-Fri/8:00-9:00, 19:30-20:30;Sat,Sun/0-24: On weekdays backup between 8 and 9 and between 19:30 and 20:30. On Saturday and Sunday the whole time.

As one can see a number can denote a day of the week (1-Monday, 2-Thuesday, 3-Wednesday, 4-Thursday, 5-Friday, 6-Saturday, 7-Sunday). You can also use the abbreviations of the days (Mon, Thues, Wed, Thurs, Fri, Sat, Sun). The times can either consist of only full hours or of hours with minutes. The hours are on the 24 hour clock. You can set multiple days and times per window definition, separated per ",". You can also set multiple window definitions. Separate them with ";".

6.2.2 Excluded files

You can exlude files with wildcard matching. For example if you want to exlude all MP3s and movie files enter something like this:

```
*.mp3;*.avi;*.mkv;*.mp4;*.mpg;*.mpeg
```

If you want to exclude a directory e.g. Temp you can do it like this:

```
*/Temp/*
```

You can also give the full local name

```
C:\Users\User\AppData\Local\Temp\*
```

or the name you gave the location e.g.

```
C_\Users\User\AppData\Local\Temp
```

Rules are separated by a semicolon (";")

6.2.3 Default directories to backup

Enter the different locations separated by a semicolon (";") e.g.

```
C:\Users;C:\Program Files
```

If you want to give the backup locations a different name you can add one with the pipe symbol ("|") e.g:

```
C:\Users|User files;C:\Program Files|Programs
```

gives the "Users" directory the name "User files" and the "Program files" directory the name "Programs".

Those locations are only the default locations. Even if you check "Seperate settings for this client" and disable "Allow client to change settings", once the client modified the paths, changes in this field are not used by the client anymore.

7 Storage

The UrBackup server storage system is designed in a way that it is able to save as much backups as possible and thus uses up as much space on the storage partition as possible. With that in mind it is best practice to use a separate file system for the backup storage or to set a quota for the 'urbackup' user. Some filesystems behave badly if they are next to fully occupied (fragmentation and bad performance). With such filesystems you should always limit the quota UrBackup can use up to say 95% of all the available space.

7.1 Nightly backup deletion

UrBackup automatically deletes old file and image backups between 3am and 5am. Backups are deleted when a client has more incremental/full file/image backups then the configured maximum number of incremental/full file/image backups. Backups are deleted until the number of backups is within these limits again.
If the administrator has turned automatic shutdown on, this cleanup process is started on server startup instead (as the server is most likely off during the night). Deleting backups and the succeeding updating of statistics can have a huge impact on system performance.

7.2 Emergency cleanup

If the server runs out of storage space during a backup it deletes backups until enough space is available again. Images are favored over file backups and the oldest backups are deleted first. Backups are only deleted if there are at least the configured minimal number of incremental/full file/image backups other file/image backups in storage for the client owning the backup. If no such backup is found UrBackup cancels the current backup with a fatal error. Administrators should monitor storage space and add storage or configure the minimal number of incremental/full file/image backups to be lower if such an error occurs.

7.3 Suitable Filesystems

Because UrBackup saves downloaded files first into temporary files and then copies them to the final location in parallel backup performance will still be good even if the backup storage space is slow. This means you can use a fully featured filesystem with compression and deduplication without that much performance penalty. At the worst the server writes away an image backup over the night (having already saved the image's contents into temporary files during the day). This section will show which filesystems are suited for UrBackup.

7.3.1 Ext4/XFS

Ext4 and XFS, are both available in Linux and can handle big files, which is needed for storing image backups. They do not have compression or deduplication though. Compression can be achieved by using a fuse filesystem on top of them such as fusecompress. There are some block-level deduplication fuse layers as well, but I would advise against them as they do not seem very stable. You will have to use the kernel user/group level quota support to limit the UrBackup storage usage.

7.3.2 NTFS

NTFS is pretty much the only option you have if you run the UrBackup server under Windows. It supports large files and compression as well as hardlinks and as such is even more suited for UrBackup than the standard Linux filesystems XFS and Ext4.

7.3.3 btrfs

Btrfs is a pretty new Linux filesystem and as such it is probably not suited for production use yet. It supports compression and in the near future will support block-level deduplication (that is already available via patches). If you do not use deduplication it should be the faster copy-on-write filesystem compared to ZFS, because it uses btrees.

7.3.4 ZFS

ZFS is a filesystem originating from Solaris. It is available as a fuse module for Linux (zfs-fuse) and as a kernel module (ZFSOnLinux). The kernel module is relatively new and thus should not be used for production purposes yet. There were licensing issues which prevented prior porting of ZFS to Linux. If you want the most performance and stability an option would be using a BSD or Debian/kFreeBSD. The ZFS in the BSD kernels is stable. In Linux you should go with zfs-fuse for the time being. The upstream Solaris ZFS has been available for some time and as such should be very stable as well. ZFS has some pretty neat features like compression, block-level deduplication, snapshots and build in raid support that make it well suited for backup storage. How to build a UrBackup server with ZFS is described in detail in section 7.4.1.

7.4 Storage setup proposals

In this section a sample storage setup with ZFS is shown which allows offsite backups via Internet or via tape like manual offsite storage.

7.4.1 Mirrored storage with ZFS

Note: It is assumed that UrBackup runs on a Unix system such as Linux or BSD. An example would be Debian/Linux or Debian/kFreeBSD with the kFreeBSD kernel being preferred, because of its better ZFS performance. We will use all ZFS features such as compression, deduplication and snapshots. It is assumed that the server has two harddrives (sdb,sdc) dedicated to backups and a hot swappable harddrive slot (sdd). It is assumed there is a caching device to speed up deduplication as well in /dev/sde. Even a fast usb stick can speed up deduplication because it has better random access performance then normal hard disks. Use SSDs for best performance.

First setup the server such that the temporary directory (/tmp) is on a sufficiently large performant filesystem. If you have a raid setup you could set /tmp to be on a striped device. We will now create a backup storage filesystem in /media/BACKUP.
Create a ZFS-pool 'backup' from the two harddrives. The two are mirrored. Put a harddrive of the same size into the hot swappable harddrive slot. We will mirror it as well:

```
zpool create backup /dev/sdb /dev/sdc /dev/sdd cache /dev/sde -m /media/BACKUP
```

Enable deduplication and compression. You do not need to set a quota as deduplication fragments everything anyways (that's why we need the caching device).

```
zfs set dedup=on backup
zfs set compression=on backup
```

Now we want to implement a grandfather, father, son or similar backup scheme where we can put hard disks in a fireproof safe. So each time we want to have an offsite backup we remove the hot swappable device and plugin in a new one. Then we either run

```
zpool replace backup /dev/sdd /dev/sdd
```

or

```
zpool scrub
```

You can see the progress of the resilvering/scrub with 'zpool status'. Once it is done you are ready to take another harddisk somewhere.

Now we want to save the backups on a server on another location. First we create the ZFS backup pool on this other location.
Then we transfer the full filesystem (otherserver is the hostname of the other server):

```
zfs snapshot backup@last
zfs send backup@last | ssh -l root otherserver zfs recv backup@last
```

Once this is done we can sync the two filesystems incrementally:

```
zfs snapshot backup@now
ssh -l root otherserver zfs rollback -r backup@last
zfs send -i backup@last backup@now | ssh -l root otherserver zfs recv backup@now
zfs destroy backup@last
zfs rename backup@last backup@now
ssh -l root otherserver zfs destory backup@last
ssh -l root otherserver zfs rename backup@last backup@now
```

You can also save these full and incremental zfs streams into files on the other server and not directly into a ZFS file system.