Open CAPWAP: WTP Update System

Donato Capitella

26/04/2010

Contents

1 Introduction

This document describes the WTP Update System I designed and developed for Open CAPWAP. These are the main goals I had in mind when designing the system:

- the update system should be independent of the underlying Unix/Linux distribution;
- the update should be possible even when the WTP structure changes radically (new binaries, new sub-directory layout, ...)
- After the update, should the new WTP not work properly, the update system has to restore an old working backup (self-healing system)

2 System Architecture

The update is distributed to the WTPs as a gzipped archive (called **CUP** - *Capwap Update Package*) with a specific format described in Section 3 and it assumes that the WTP working directory contains the binaries and the configuration files (logs should be saved elsewhere). When an update has been received, the WTP shuts down and starts the **WUA** (*WTP Update Agent*) which takes care of carrying out the update procedure and, in the end, starts the new WTP.

2.1 WTP Version Numbers

The WTP version consists of three numbers separeted by dots (eg. 0.93.2):

- **MAJOR NUMBER**: the main release number
- **MINOR NUMBER**: indicates bug fixes and minor new features
- **REVISION NUMBER**: minor bugfixes or configuration changes

During development, the WTP version can be modified in the header file WUM.h.

2.2 Update Messages and WTP States

The WTP Update System, according to the received messages, passes from one state to another. There are three states:

- **WAIT**: the WTP is waiting for update requests
- **BUSY**: the WTP is busy receiving the fragments of an update
- **READY**: the WTP has successefully received all the fragments of an update and is ready to launch the WTP Update Agent.

The update system uses 10 new messages to carry out an update procedure. These messages are:

- **WTP_VERSION_REQUEST**: the AC asks a WTP its version. The WTP answers with a **WTP_VERSION_RESPONSE**.
- **WTP_UPDATE_REQUEST**: the AC asks the WTP if it can execute an update. In the request the update version and the required disk space are specified. The WTP answers with a **WTP_UPDATE_RESPONSE**. Moreover, if it can execute the update, the WTP creates a temp file which will be used to store the CUP and passes to the BUSY state, meaning that it is accepting update fragments.
- **WTP_CUP_FRAGMENT**: this message is used to deliver a fragment of the CUP to the WTP. The CUP has to be divided into fragments before being sent and each fragment is smaller than 4k. The WTP acknowledges the fragment with a **WTP_CUP_ACK**. When all fragments have been received, the WTP Update System passes to the READY state.

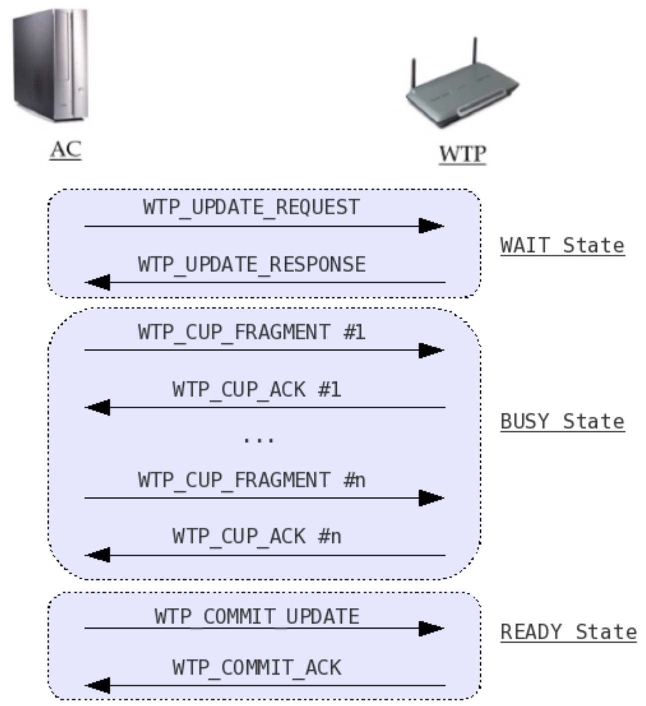


Figure 1: Capwap Update Messages

- **WTP_COMMIT_UPDATE**: the AC tells WTP to shutdown and start the WTP Update Agent. The WTP answers with a **WTP_COMMIT_ACK** before shutting down.
- **WTP_CANCEL_UPDATE_REQUEST**: the AC asks the WTP to cancel an update. This works if the WTP is in the BUSY or READY state, before a WTP_COMMIT_UPDATE has been received. The WTP, after cleaning the temp files, goes back to the WAIT state and answers with a **WTP_CANCEL_UPDATE_RESPONSE**.

These ten messages are packed as Configuration Update Requests/Responses of *`Vendor Specific Payloads`* type: this type of messages was introduced in the CAPWAP Protocol to allow vendors to extend the protocol with new features, such as this update system. Figure 1 shows a message exchange between the AC and the WTP during a successful update procedure.

Figure 2 shows a finite state machine which describes the transitions among the states of the update system; each edge is labelled with the request message which triggers the transition.

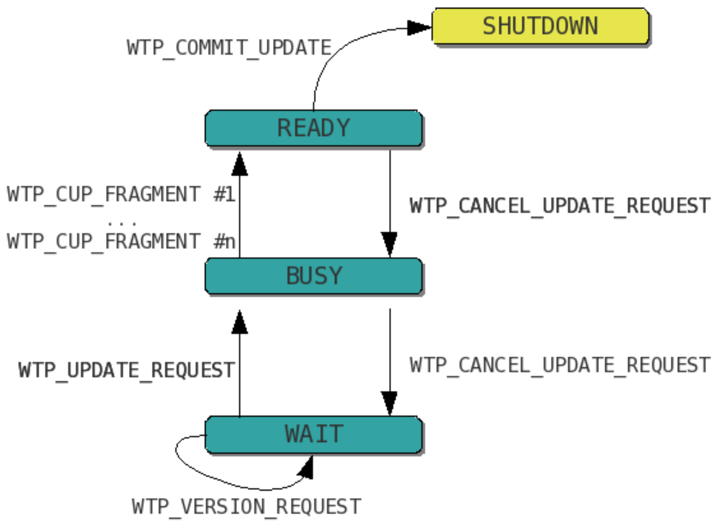


Figure 2: Capwap Update System States

3 Creating a CUP

In the following section I describe what a CUP is and how to build one.

3.1 CUP Structure

The update packages, CUPs from now on, are distributed as gzipped archives with the following structure:

- `update.cud`: the CUD (*Capwap Update Descriptor*), an ASCII text file describing the update.
- `WTP/`
  - new or updated files (binaries, configuration files, ...)
- `scripts/`
  - pre and post update scripts

The following subsection describes the format of the CUD.

3.2 CUD: Capwap Update Descriptor

The CUD is an ASCII text file that describes an update. It has the very simple 'option: value' format. The options implemented so far are the following:

- **Version**: the version shipped with the update (`MAJOR.MINOR.REVISION`)

- **PreUpdate**: script to execute before the update takes place (pathname relative to the script subdir)
- **PostUpdate**: script to execute after the update has taken place (pathname relative to the script subdir)

The option names are case-insensitive. The only required option is 'Version'.

The pre and post update scripts are provided to costumize the update procedure as much as possible; for example, they can be used to edit and convert existing configuration files to new formats. These scripts have access to two shell variables: *$WTP_DIR* and *$CUP_DIR*. $WTP_DIR is the WTP working directory while $CUP_DIR holds the pathname of the temp directory in which the CUP was unpacked.

Here follows an example of update.cud:

update.cud

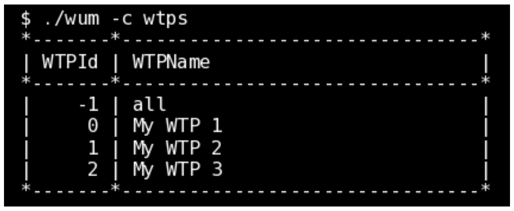
```
Version:  0.94.4
PreUpdate:  pre.sh
PostUpdate:  post.sh
```

4 Using wum

Wum (WTP Update Manager) is the command line tool used to send update requests to the WTPs. Of course it is executed on the AC machine and interacts with the AC Interface.

4.1 Getting a list of the WTPs

By default, wum tries connecting to a local AC listening on port 1235. This behaviour can be changed using the options -a <address> and -p <port>. To get a list of all the active WTPs connected to the AC you can use the '-c wtps' switch (the c stands for 'command' and 'wtps' is just one of the supported commands):



```
$ ./wum -c wtps
*-------*-------------------------------*
| WTPId | WTPName                       |
*-------*-------------------------------*
|    -1 | all                           |
|     0 | My WTP 1                      |
|     1 | My WTP 2                      |
|     2 | My WTP 3                      |
*-------*-------------------------------*
```

Figure 3: List of wtps

4.2 Checking WTP version

In order to check the version of a WTP you can use the 'version' command. To specify which WTP you are interested in you can use either the -w or -n switches that let you input a comma-separated list of WTP ids (-w) or a comma-separated list of WTP names(-n). For exmaple, if you want to know which version of the WTP software is running on 0 and 2, you can use either of the following commands:

- `wum -c version -w 0,1`
- `wum -c version -n 'My WTP 1','My WTP 2'`

You can also use -1 or 'all' if you want to contact all the WTPs.

4.3 Sending an update

To send an update to a list of WTPs you can use the 'update' command. This comand, in addition to the WTP list, requires a cup file to be specified using the -f option. Here's an example in which we send an update to all the WTPs:

- `wum -c update -n 'all' -f capwap0.93.4.cup`

4.4 Cancel an update: if something goes wrong

If something goes wrong while sending an update, the WTP Update System might remain in the BUSY or READY state. If that's the case, the WTP won't accept any further update request. If such a scenario occurs, you need to send a WTP_CANCEL_UPDATE_REQUEST in order to bring the WTP back to the WAIT state and retry the update procedure. This is done using the 'cancel' command.

5 WTP Update Agent

When the WTP receives a WTP_COMMIT_UPDATE message, if in the correct state (i.e. READY), it answers with a WTP_COMMIT_ACK, shuts down, and starts the WTP Update Agent. This piece of software takes care of carrying out an update session which is divived into 3 main stages.

- **Stage 1**: Preparation
  - unpacks the CUP into a temp dir (/tmp/cup.unpack)
  - backups the current WTP working directory (/tmp/cup.backup)
- **Stage 2**: Actual update
  - executes the pre-update script

  - copies into the WTP working directory the new files shipped with the update (i.e. the content of /tmp/cup.unpack/WTP)
  - executes the post-update script

- **Final Stage**: Cleaning up
  - starts the new WTP
  - if everything is OK, the WUA cleans the temp files and exits.

If anything goes wrong during the first state, the WUA simply deletes the temp files and restarts the old WTP. If anything happens during the second stage, prior to deleting the temp files, the WUA restores the backup to ensure that a working version of the WTP is preserved.

A log of every update session is mantained in **/var/log/wua.log**.