Universal Cloud Downloader

Internet Of Things Series - Marzo 2020
MAC srl - Ing.F.Castelli - rev 1.1

Host Implementation Guide

Universal Cloud Downloader

Universal Cloud Downloader

Overview

UCD e' una Architettura Cloud per la notifica e la distribuzione di nuove release files o moduli software verso un qualunque Dispositivo, autenticato e con credenziali valide, dotato di scarse risorse di buffering.

La nuova release viene scaricata a blocchi rispettando la dimensione massima richiesta dal Dispositivo per ciascun blocco.

Il Dispositivo puo' in qualunque momento ripetere la procedura, riprenderla a sua convenienza o abortirla. Il check periodico dell'aggiornamento e' svolto autonomamente dalla logica interna al dispositivo. La massima dimensione del file e' fissata a 4GB, e la dimensione massima richiedibile per il singolo blocco e' di 65535 bytes(<64KB). Il file puo' essere indifferentemente di natura binaria o di testo, e di qualunque genere: memoria firmware, immagini grafiche, testi, configurazioni, file multimediali, ...

L'Architettura Cloud consiste in:

- un Gestore di accessi con database dei dispositivi e degli operatori abilitati
- un Portale per la registrazione Applicazioni e pubblicazione e upload delle Releases
- una Interfaccia API REST per la gestione procedure di segnalazione e upload
- un File Manager per lo storage e mantainance delle release e e dei files
- un Motore Middleware(DWLOADER ENGINE) per segnalazione e download delle nuove Release per le Applicazioni registrate
- un canale di comunicazione sicura (TSL 1.2) con ausilio di Certificati per una comunicazione end-to-end cifrata e autenticata

L'Architettura generale si completa di un protocollo Seriale per HOST di facile implementazione per consentire una immediata integrazione nel firmware: `DWLOADER HOST PROTOCOL`.

Applicazione

Ogni contesto applicativo di una stessa tipologia di Dispositivi viene definito: `Applicazione`.

In ambito di una Applicazione possono essere presente un numero illimitato di files di qualuqnue natura con associato un versioning convenzionale. Questi file sono chiamati: `Release`.

L'Architettura puo' gestire qualunque numero di Applicazioni.

Es. Un ambito Applicativo puo' essere la gestione e manutenzione delle release di memoria binarie legate a Dispositivi HOST di una specifica macchina.

Identificativo dispositivo

Si intende con DISPOSITIVO l'intera macchina, composta da:

- *HOST* (microprocessore gestore applicazione) e un
- Modulo di Comunicazione CLOUD detto *DOWNLOADER* (responsabile connessione wifi e del collegamento ai servizi Universal Cloud Downloader).

Ogni dispostivo di fabbrica ha univoco `DeviceID` e `LoginKey` (per accesso ai servizi Cloud), entrambi mantenuti dal modulo DOWNLOADER. L'accesso ai servizi Cloud e' garantito dal LoginKey che per qualunque motivo il Gestore Servizi puo' rigettare disattivando l'utilizzo del Cloud al dispositivo.

Il modulo DOWNLOADER si interfaccia all'Architettura Cloud, identificantosi con le proprie credenziali e utilizzando le API REST messe a disposizione.

Il modulo HOST, che non ha connessione Cloud, e in genere costituito da un microcontrollore o microprocessore da scarse risorse di memoria, colloquia con il modulo DOWNLOADER tramite il seguente protocollo seriale: DwLoader Host Protocol.

DwLoader Host Protocol

Protocollo di comunicazione tra HOST-DOWNLOADER

Il dispositivo che richiede il download dei file remoti e' detto HOST, ed e' Master: ha l'iniziativa nella comunicazione, il dispositivo che opera come Slave e' detto DOWNLOADER e ha il compito di riceve i blocchi del file remoto che giungono dal Cloud e confezionarli per l'HOST. Il Cloud gestisce la distribuzione ed il versioning dei file (CURRENT RELEASE) relativi alla terna {APPLICATION,RELEASE,ESTENSIONE} richiesta.

Il DOWNLOADER fornira'

- segnalazione di Release piu' recenti,
- la conferma dell'allineamento delle Release
- oppure la Risposta Negativa a richieste di Release non presenti

HOST, sulla base della {APPLICATION,RELEASE,EXTENSION}, puo' scaricare qualunque tipologia di file. Applicazione tipica e' lo scarico di una nuova release di firmware (si puo' forzare anche il downgrade di firmare, impostando la release desiderata come CURRENT RELEASE). HOST segnala al DOWNLOADER la sua capacita' di immagazzinamento per i blocchi da ricevere ed il DOWNLOADER fornira' blocchi di quella dimensione od inferiore.

Comando CHECK_RELEASE

Regolarmente l'HOST controlla la presenza di una nuova release con questo comando.

Il comando ha la seguente sintassi:

```
;app=<application>&rel=<release>&ext=<estensione>:<crc16>

ove
<application>: stringa ASCII di min 3, max 13 caratteri alfanumerici
<release>    : stringa ASCII di min 3, max 13 caratteri alfanumerici
<extension>  : stringa ASCII di 3 caratteri alfanumerici
<crc16>      : 4 caratteri NIBBLEASCII (maiuscoli) rappresentante il
               CRC16_CCITT_ZERO(*) di tutto il messaggio escluso l'header ;
```

(*) CRC16_CCITT_ZERO puo' essere verificato online al seguente indirizzo: CRC calculator *(http://www.sunshine2k.de/coding/javascript/crc/crc_js.html)*

L'Host riceve dal DOWNLOADER, che si connette in modo autenticato al servizio Universal Cloud Downloader, la seguente risposta:

RISPOSTA POSITIVA

```
;OK:2        Aggiornamento non necessario: Release allineata.
```

oppure

```
;OK:1<filename>:<crc16>   nuova release <filename> disponibile sul Cloud

ove
<filename>  : stringa ASCII alfanumerica  della nuova release
              da recuperare. Il formato risulta:
              <application>_<release>.<extension>  (min 11, max 31 caratteri)
<crc16>     : 4 caratteri NIBBLEASCII (maiuscoli) rappresentante il
             CRC16_CCITT_ZERO di tutto il messaggio escluso l'header ;
```

oppure

```
;OK:3        Applicazione non trovata
```

oppure

```
;OK:0        Risposta assente(timeout) o ritornato errore dal Cloud
             Riprovare.
```

oppure

```
;OK:5        Non autorizzato
```

RISPOSTA NEGATIVA

```
;KO:<nack_code>   Codice errore, per i codici vedi la sezione relativa
```

Comando GET_SIZE

Ricevuto il filename di una nuova release, l'HOST richiede le dimensioni col seguente comando:

```
;file=<filename>&block_size=<bytes>:<crc16>

ove
<filename>   : stringa ASCII del file ritornato dal CHECK RELEASE
<bytes>      : stringa ASCII rappresentante un intero (max 65535)
               corrispondente al
               blocco massimo gestibile dall'HOST
<crc16>      : 4 caratteri NIBBLEASCII (maiuscolo) rappresentante il
               CRC16_CCITT_ZERO di tutto il messaggio escluso l'header ;
```

RISPOSTA POSITIVA

```
;OK:1#blocks=<numero_blocchi>,filesize=<intero_lungo>:<crc16>

ove
<numero_blocchi> : numero blocchi di dimensione <block_size>
                   (o eventualmente inferiore per ultimo blocco)
                    di cui e' composto il file richiesto
<intero_lungo>   : dimensione totale in bytes del file richiesto
```

oppure

```
;OK:3        file non presente
```

oppure

```
;OK:0        Risposta assente(timeout) o ritornato errore dal Cloud
             Riprovare.
```

oppure

```
;OK:5        Non autorizzato
```

RISPOSTA NEGATIVA

```
;KO:<nack_code>  Codice errore, per i codici vedi la sezione relativa
```

Comando GET_BLOCK

L'HOST puo' richiedere di scaricare i blocchi del file richiesto col seguente comando:

```
;get=<filename>&block_size=<bytes>&block_index=<indice_corrente>:<crc16>

ove
<filename>   : stringa ASCII (minuscolo) del file ritornato da CHECK_RELEASE
<bytes>      : intero (max 65535) rappresentante il blocco massimo
               gestibile dall'HOST
<indice_corrente> : intero lungo rappresentante l'indice corrente
                    del blocco da scaricare
                    (indice del primo blocco parte da 0)
<crc16>      : 4 caratteri NIBBLEASCII (maiuscolo) rappresentante il
               CRC16_CCITT_ZERO di tutto il messaggio escluso l'header ;
```

RISPOSTA POSITIVA

```
;OK:1<NIBBELASCII_block_index><NIBBELASCII_block_len><blob>:<crc16>

ove
<NIBBELASCII_block_index>: 4 caratteri NIBBLEASCII (maiuscolo)
                           rappresentate indice del blocco richiesto
<NIBBELASCII_block_len>  : 4 caratteri NIBBLEASCII (maiuscolo)
                           rappresentate la dimensione del blocco dati a
                           seguire (uguale al <block_size> richiesto
                           o eventualmente minore in caso di ultimo blocco)
<blob>       : blocco dati binario richiesto
<crc16>      : 4 caratteri NIBBLEASCII (maiuscolo) rappresentante il
               CRC16_CCITT_ZERO di tutto il messaggio escluso l'header ;
```

oppure

```
;OK:4        indice eccede le dimensioni del file
```

oppure

```
;OK:0        Risposta assente(timeout) o ritornato errore dal Cloud
             Riprovare.
```

oppure

```
;OK:5        Non autorizzato
```

RISPOSTA NEGATIVA

```
;KO:<nack_code>
```

Codici di Ack

Codici che seguono il marker `;OK:` in caso di Risposta Positiva:

```
ACK_ABORTED             '0'    timeout o errore cloud link
ACK_WELL_DONE           '1'    comando eseguito, segue la risposta e il crc
ACK_NO_ACTION           '2'    release allineata
ACK_NOT_FOUND           '3'    Applicazione/Release richiesta non trovata
ACK_OVER_LIMIT          '4'    indice_blocco eccede i blocchi utili
ACK_NOT_AUTH            '5'    non autorizzato
```

Codici Errore

```
NACK_WIFI_NO_LINK       '0'    DOWNLOADER non connesso all`Access Point
NACK_HEADER_SEMICOLON   '1'    errore sintassi carattere di start
NACK_SYNTAX_ERROR       '2'    errore sintassi
NACK_INVALID_SIZE       '3'    errore dimensione
NACK_END_MARKER         '4'    errore carattere di stop
NACK_CRC_ERR            '5'    crc errato
```

Tempistica transazione

La comunicazione e' di natura asincrona ed e' in genere utilizzata un tecnologia UART con i seguenti parametri di default: 115200,N,8,1 (adattabile alle esigenze dell'HOST).

La durata della transazione, avendo una connessione remota al Cloud, ha delle latenze conseguenti. L'invio dei marker `;OK:` `;KO:` sono onorati in frazioni di secondo (<200ms) ma il codice di Ack occorre attenderlo sino a 10 secondi nei casi peggiori (Timeout, Cloud errors, Internet down). Tempi medi transazione, dipende dalla dimensione richieste dei blocchi e latenza connessione internet.

Es: per block_size di 2KB: il blocco e' ricevuto dal Cloud mediamente entro 3 secondi.

Esempi

A seguire alcuni esempi di scambio dati tra HOST e DOWNLOADER

```
CHECK RELEASE (nuovo aggiornamento disponibile)
<==   ;app=eagleone&rel=mf000003&ext=mot:9ABA
==>   ;OK:1eagleone_mf000004.mot:7F61


CHECK RELEASE (nessun aggiornamento)
<==   ;app=eagleone&rel=mf000004&ext=mot:8640
==>   ;OK:2


GET_SIZE  (file non presente)
<==   ;app=testnotfound&rel=mf000004&ext=mot:41BD
==>   ;OK:3


GET_SIZE  (info su file richiesto)
<==   ;file=eagleone_mf000004.mot&block_size=2048:9BEF
==>   ;OK:1#blocks=49,filesize=100029:7A3E


GET_BLOCK  (recupero blocco numero 0 di 2048 bytes)
<==   ;get=eagleone_mf000004.mot&block_size=2048&block_index=0:A7A3
==>   ;OK:100000800S00E0000435058335F426F6F6D6F7404
      S315FFFA010043705873426F4F74486541644FF1FFFF6E
      S315FFFA011040000000000000720C1090000231CFBFF76
      S315FFFA0120010A000000000248014201008 7B55FFF9D
      S315FFFA0130024C0100000000481133000032 6DE6FF61
      S315FFFA0140137F010000000000D10902004DFFFBFEFC
      ...
      S315FFFA0360A918659E1E29C3E8B98C6D6B73327A0F8D
      S315FFFA0370B314BA804604:0EE1

GET_BLOCK  (recupero blocco numero 90 di 2048 bytes non esistente)
<==   ;get=eagleone_mf000004.mot&block_size=2048&block_index=90:804E
==>   ;OK:4
```

Appendice: crc16 C source code

Di seguito una implementazione del calcolo CRC16_CCITT_ZERO, in linguaggio C, senza utilizzo di librerie esterne

```
static const unsigned short CRC_CCITT_TABLE[256] =
{
 0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
 0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
 0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
 0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
 0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
 0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
 0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
 0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
 0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
 0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
 0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
 0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
 0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
 0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
 0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
 0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
 0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
 0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
 0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
 0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
 0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
 0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
 0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
 0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
 0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
 0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
 0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
 0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
 0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
 0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
 0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
 0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};


unsigned short Calculate_CRC_CCITT(const unsigned char* buffer, int size)
{
```

```
    unsigned short tmp;
    unsigned short crc = 0;

    for (int i = 0; i < size; i++)
    {
        tmp = (crc >> 8) ^ buffer[i];
        crc = (crc << 8) ^ CRC_CCITT_TABLE[tmp];
    }

    return crc;
}
```

Appendice: C source code reference

A seguire il link di una implementazione in C source code del protocollo per HOST e Applicazioni BOOTLADER (`C working sample`)

DWLOADER-C-SOURCE-WORKING_SAMPLE.zip *(http://dwloader.meshgrid.it/DWLOADER-C-SOURCE-WORKING_SAMPLE.zip)*