CUCM 11.5.x TFTP Scale Architectural Improvements

Contents

Introduction

This document in about the Trivial File Transfer Protocol (TFTP) scale architecture feature implemented as a part of Cisco Unified Communication Manager (CUCM) version 11.5 the newest uplift to CUCM. This is purely a engineering feature in order to improve TFTP service with respect to memory usage and how it serves the configuration and static files. The business logic remains the same and there is no impact with respect to other services provided by TFTP.

Background Information

Reasons why this improvement was required and incorporated

Problem with current Design

- The logic of how TFTP serves the configuration files hasn't been changed for a long time.
- Pre 11.5, TFTP service builds the configuration files and caches all the configuration files in-memory.
- With more capacity added to CUCM with respect to number of phones supported, the memory foot print of TFTP service linearly increased.
- Future roadmaps have the requirement of additional capacity for phones in order to be implemented in CUCM.
- Hence, address the increas of memory foot print of TFTP service becomes important.

Service start up time

- In a medium to large deployments with 20k to 40k phones configured.
- When a change is made that affects all the phones, TFTP builds all the affected configuration

files and rebuilds it cache.
- This increases the time taken for the TFTP service to start.
- At the time when phones request for configuration file a busy response is sent to the phone.

Feature Overview

The new feature implemented addresses the above two problems by a cache-less design and build the configuration file on-demand. When a request is sent from phone, the TFTP service builds the configuration file on the fly and serves it to the phone in real time. It won't cache the configuration file in-memory which in turn it reduce the service start time and the memory footprint of the TFTP service.

Design changes

Design changes done fall under two categories namely 'Connection Management' and 'Configuration File Generation'. The below table details the changes done under each category.

Connection Management		Configuration File Gener
HTTP	**TFTP**	Framework added for on-de
Network Service layer are designed to use SDL in order to handle all TCP connections	No changes where phones request the configuration files over UDP	build and signed configura files

Performance Improvements

Below are the performance improvements achived with implementation of this new feature.

- Significant reduction in memory footprint of TFTP service
- Memory footprint is around 600 MB for the TFTP service
- Service start time is lesser since the files are not cached
- The service start time is independent of the number of phones deployed in the system

Performance Figures

	No. of Phones	Time take in Pre 11.5 version	Time Taken in 11.5 version
Service Start Time	20000	3 Minutes 38 Seconds	0 Minutes 19 Seconds
Files Served over HTTP	20000	7 Minutes 24 Seconds	4 Minutes 06 Seconds
Files Served over TFTP	20000	5 Minutes 36 Seconds	4 Minutes 11 Seconds

Note: The above numbers are not just from one test run but is a average of several test runs.

Log Analysis :

Devices Used :

CUCM version 11.5.1.10000-6

Cisco IP Communicator version 8.6.2

Configuration file request over HTTP in pre 11.5

Request from phone for the configuration file

```
00593088.000 |21:58:11.698 |AppInfo  |   TID[da900b70] HTTPEngine::getRequest(),
[0xa0d6c90~7~10.65.64.132~54462] INFO:: socket(12), ReqTimeout[60],
Request[GET /SEP000C29ED3D88.cnf.xml HTTP/1.1
```

Since all files are cached after built, TFTP finds the cached configuration file

```
00593097.000 |21:58:11.698 |AppInfo
|CReqContext::FindAndServe(1)[0xa0d6c90~7~10.65.64.132~54462]
,[(SEP000C29ED3D88.cnf.xml),(6779),(0xf388c2a8)] found in config cache
```

The configuration file is successfully served to the phone

```
00593102.000 |21:58:11.698 |AppInfo  |
HTTPEngine::sendResponse[0xa0d6c90~7~10.65.64.132~54462]
FileName[SEP000C29ED3D88.cnf.xml], Version[HTTP/1.1], Size[6779] 00593103.000 |21:58:11.698
|AppInfo | HTTPEngine::sendResponse[0xa0d6c90~7~10.65.64.132~54462]
INFO:: [85][HTTP/1.1 200 OK
```

Configuration file request over HTTP in 11.5

Request from phone for the configuration file

```
00000510.003 |21:47:40.683 |AppInfo  |   HTTPConnection::wait_SdlDataInd Printing the
HTTPRequest :
msgBuffer size [148] --: GET /SEP000C29ED3D88.cnf.xml HTTP/1.1
```

ServeFile process sends the signal 'FileRequest' to ServeDynamicFile

```
00000511.010 |21:47:40.683 |AppInfo  |   ServeFile::wait_FileRequest Sending the
FileRequest signal to ProcessServeDynamicFile process 00000511.011 |21:47:40.683 |AppInfo |<--
ServeFile::wait_FileRequest 00000512.000 |21:47:40.683 |SdlSig |FileRequest |wait
|ServeDynamicFile(1,600,25,1) |ServeFile(1,600,24,1) |1,600,14,4.3^*^* |*TraceFlagOverrode
```

Since cacheless design is implemented, you see that TFTP builds the configuration file

```
00000512.027 |21:47:40.684 |AppInfo  |TFTPList::GetSupportsFMT(), Pkid[9e9cb809-df9f-4bce-8a41-
37cd5f7e4d21] Name[SEP000C29ED3D88] Class[1] Product[30041] Model[30016] Protocol[0],
DevProfile[0] SUPPORTs[2], Value[2]

00000512.028 |21:47:40.684 |AppInfo  |<--TFTPList::SelectByDeviceID[0,0]

00000512.029 |21:47:40.684 |AppInfo  |   ServeDynamicFile::wait_FileRequest
Build Config file for Device [SEP000C29ED3D88]
```

ServeDynamicFile process sends the signal 'FileResponse' to ServeFile

```
00000512.091 |21:47:40.686 |AppInfo  |<--ServeDynamicFile::wait_FileRequest
00000513.000 |21:47:40.686 |SdlSig   |FileResponse |wait
|ServeFile(1,600,24,1) |ServeDynamicFile(1,600,25,1) |1,600,14,4.3^*^* |*TraceFlagOverrode
00000513.002 |21:47:40.686 |AppInfo | ServeFile::wait_FileResponse File
Response signal received by ServeFile process
```

Requested file is sent to the phone

```
00000514.001 |21:47:40.686 |AppInfo  |-->HTTPConnection::wait_FileResponse
00000514.002 |21:47:40.686 |AppInfo  |   HTTPConnection::wait_FileResponse Requested
```

file FOUND... Sending file Response 00000514.003 |21:47:40.686 |AppInfo |<-- HTTPConnection::wait_FileResponse