\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Configuring a local VoIP service

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Author*:

Gabriel Rosas



Purpose

When I want to talk to an English friend remotely from Seattle, I can easily call them and receive near instantaneous audio transmission, if I am connected to the internet. **Voice over Internet Protocol** (VoIP) allows voice to transmit over the internet instead of the old-fashioned copper phone lines. With the internet expanding ever so vastly into the average person’s life, especially during the pandemic, it only seemed natural to take a gander behind the scenes of VoIP.

Background Information

Why do we use VoIP?

Originally, copper landlines were used as the media transmitting incoming and outgoing calls. Since then, advancements have been made in networking allowing voice to be sent as packets over the internet. While some might relate VoIP to on-premises hardware – the classic, physical phones typically used in offices for marketing – other software, such as Skype or Discord, both incorporate VoIP to manage calls.



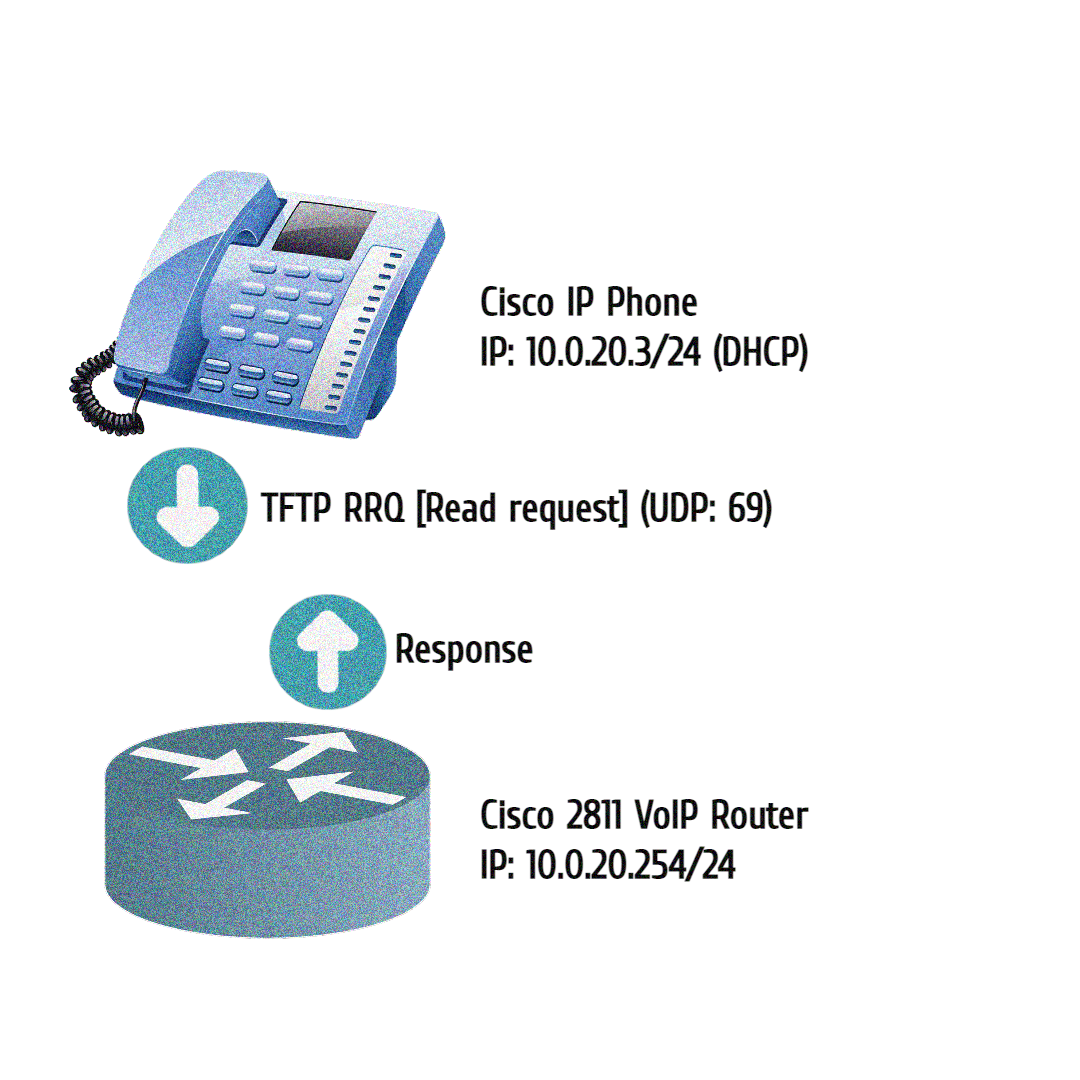
People opt for VoIP since a connection to the internet is the only medium needed for conference. Other infrastructure, such as extra copper phone lines, is not necessary. Unlike landline phones, VoIP is capable of high-definition phone calls.

What is the Cisco Unified Communications Manager?

The Cisco Unified Communications Manager (CUCM) functions as a controller for IP-based communications. In other words, calls begin, are maintained, and end with the CUCM. While the CUCM can manage calls, it also hosts files for IP phones on the network for when they initially boot up. There are two ways of configuring VoIP with the CUCM: with the full CUCM that runs on a server, or the CUCM Express (CUCME) which runs on certain routers. The full CUCM has many services, such as call manager, dhcp service, tftp, and service analyzers. The CUCME is ported down to be runnable in a router. In this lab, I host a call between two IP phones using the CUCME running on the Cisco 2811 Integrated Services Router.

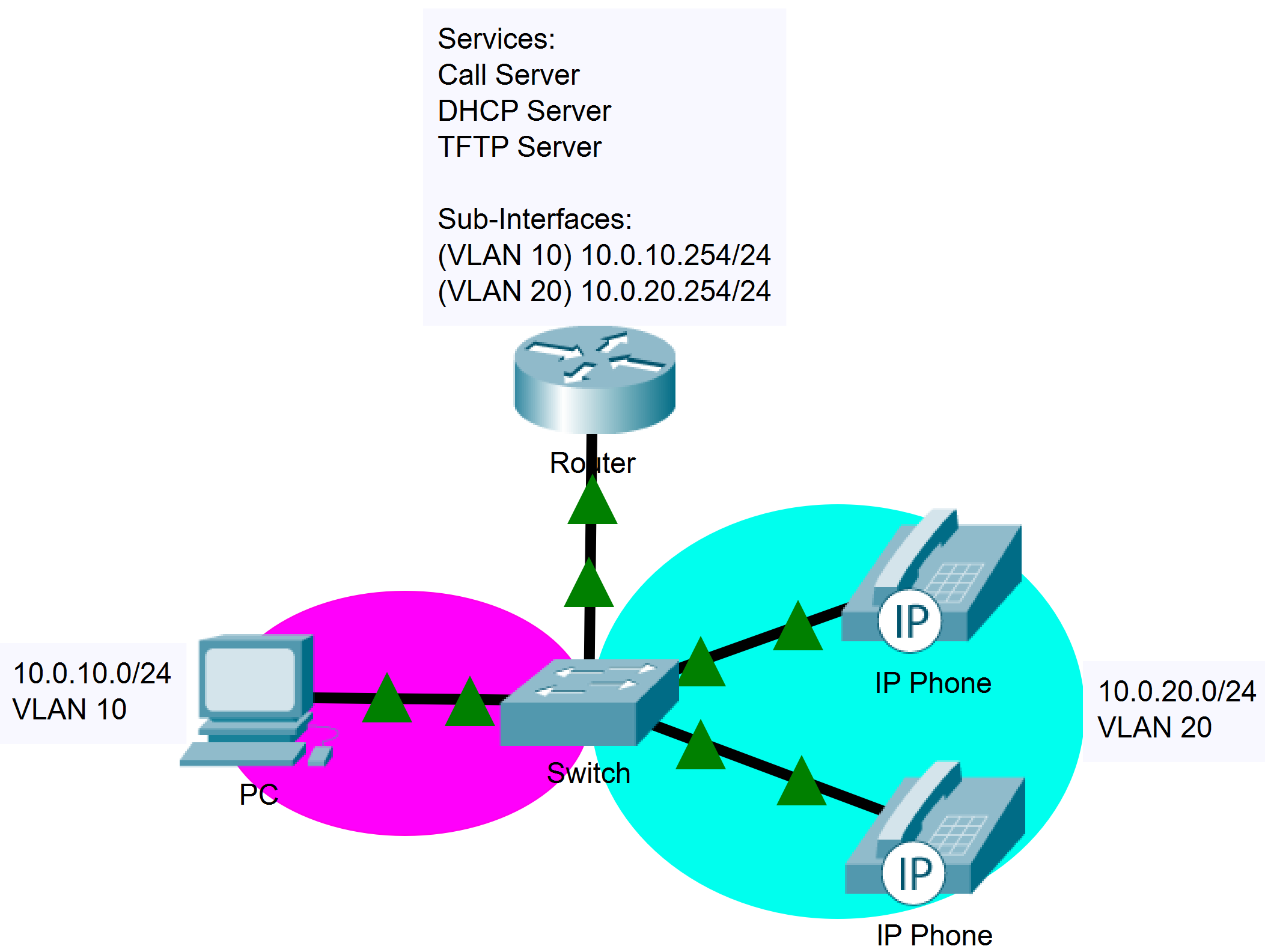
Trivial File Transfer Protocol

Trivial File Transfer Protocol (TFTP) allows a client to obtain a file from or upload a file to a remote host. Unlike File Transfer Protocol (FTP), which uses TCP ports 20 and 21, TFTP uses UDP port 69 to establish network connections. While some may question the application of UDP, the faster but less secure method of data communication, the reasoning becomes clearer if one considers the *triviality* of TFTP. TFTP is much simpler than FTP. Files hosted by a TFTP server are generally quite small; you are more likely to find TFTP used on a LAN while FTP is implemented on a WAN; network engineers use TFTP to distribute software across a LAN because packet loss is unlikely.



Why do we care about TFTP? Cisco SCCP IP phones attempt to download software on boot that we need to host. This process is known as bootstrapping. Bootstrapping allows machines that are new to the network, or machines that have lost everything, to download software files which get them up and running. Cisco routers let engineers host files using TFTP. However, a Cisco router is not a fully functioning TFTP server, it can only serve files for download. Any file in the router’s flash can be hosted, including the IP phone files we need.

Network Diagram



Process

The objective for this lab was to get two IP phones making and receiving calls from each other. Without knowing anything about VoIP or the CUCM, my first goal was to construct a topology with two IP phones and worry about the configurations later. I’ve found that, like math, visually plotting out the data is a great first start for any project. To construct the topology, I needed a basic grasp on how VoIP works with the CUCM.

I ended up referencing a YouTube video which provided general configurations for what I wanted to achieve. Following the video, I was able to design the topology above and create points of each major step needed to get VoIP running.

Steps

* Vlans
  + Create two vlans – one for VoIP and one for the computers.
  + Enable spanning tree portfast on the VoIP interfaces. While it is not necessary, portfast will reduce the delay a phone has when receiving its IPs and configuration files.
  + Set up the respective vlans and sub-interfaces on a VoIP router.
* DHCP
  + Create two DHCP pools – one per vlan. For the VoIP pool, define a DHCP option 150 address (the IP of your TFTP interface, likely the VoIP sub-interface address).
* CUCM (telephony-service)
  + Load an IP phone file.
  + Define the max ephones of the network.
  + Define the max number of virtual voice ports (*ephone-dn*).
  + Define the call server address (*ip source-address*).
  + Create a system message (optional).
* Ephone
  + Manually define each ephone connected to the network.
  + Define the mac address.
  + Define what some buttons on the IP phone do.
* TFTP
  + Host local IP Phone files on the router using *tftp-server*.
  + IP phones will request files from the interface defined in DHCP option 150.
* Create CNF files
  + Go back into the CUCM and create files (*create cnf-files*).

Problems

After creating the vlans and setting up DHCP, I was left with the new content: Telephony configuration. My DHCP service was working across both vlans, so up until this point, I had no problems. I defined the call server address in *telephony-service* and hosted *tftp-server* IP phone files in global config. I decided to configure one ephone before configuring the other, assuming that both ephones would run by simply tweaking a working config. The difference between *ephone* and *ephone-dn* initially confused me, but I eventually realized ephone dial numbers (*ephone-dn*) “attach” to ephones. Once I created the ephone config by assigning a *mac-address* and an *ephone-dn*, the respective IP phone booted and received files. All I needed to do was replicate that config for my other IP phone.

Though I used the same ephone configuration – except for a different *mac-address* and *ephone-dn* – my second IP phone ran into an error. I kept receiving the message “TFTP Files Not Found”. It was strange how one ephone was able to locate the files while the other was not, considering the TFTP server hosted the same files. Both IP phones were the same model, *Cisco 7940s*, so that could not be the issue.

Running through forums on the internet, I noticed a pattern in most *tftp-server* configurations: multiple files were listed. Some even had “alias’s” that were directly referenced in *telephony-service*. For example, the command *tftp-server flash:P00308000500.bin alias P00308000500* in global configuration mode is then referenced by the command *load 7960-7940 P00308000500* in *telephony-service* mode. After adding more of these references to files in my configuration and using the *debug tftp events* command to show what files the IP phone is trying (but failing) to grab, I got the second IP phone to boot.

Lab Commands

|  |  |
| --- | --- |
| **Command** | A statement necessary for a configuration to work, denoted in bold |
| **[*Argument*]** | An argument necessary for a command to function, denoted in bold italics. |
| *Optional-Statement*  *<Optional Argument>* | An optional argument or statement, not necessary for a command to function, denoted in italics |

// IPv4 DHCP Configuration

Router(config)# **ip dhcp exclude-address [*initial ip*]** <*final ip*>

* Set an IP or range of IPs to exclude from the pool

*If the network administrator so chooses to exclude a range of IP addresses, the range would be from the* Initial IP *to the* End IP*, inclusive. The* End IP *argument is not necessary when excluding only one IP. Excludes are typically reserved for pre-configured static IP addresses, for example, interfaces on the router.*

Router(config)# **ip dhcp pool [*pool name*]**

* Creates a pool for distributing routing information

*Dynamic Host Configuration Protocol (DHCP) automates the assignment of IP addresses to devices on the local network. A DHCP pool is used to define the range of IP addresses that the server will divvy out to clients.*

Router(dhcp-config)# **network [*network address*] [*subnet mask*]**

* Configures a pool that distributes the specified subnet

Router(dhcp-config)# **default-router [*ip*]**

* Sends the specified default gateway to clients

Router(dhcp-config)# **option 150 ip [*call server address*]**

* Advertises the address of the TFTP server where IP phones will request files

*Cisco IP phones download their files from a TFTP server. Once an IP phone receives DHCP information, gaining an IP, it will then attempt to download the files.*

// VLAN Configuration

Switch(config)# **vlan [*id*]**

* Create a vlan with specified *id*

*A Virtual Lan (vlan) is used to partition groups of devices on a switched network.*

Switch(config-if)# **switchport mode access**

* Configure an interface to be in access mode

*Only one vlan is permitted across an access mode interface. Devices connected to an access-configured interface are usually end devices, such as clients.*

Switch(config-if)# **switchport access vlan [*id*]**

* Configure an interface to be part of a specified vlan

*To configure the same command access across multiple interfaces, use the command* ***interface range [interface] [start-end id]****.*

Switch(config-if)# **switchport trunk encapsulation dot1q**

* Configures the interface to use IEEE 802.1Q encapsulation on frames when the interface is configured as a trunk

*The user must define the encapsulation before setting an interface as a trunk.*

Switch(config-if)# **switchport mode trunk**

* Configure an interface to be a trunk

// Router Sub-Interface Configuration

Router(config)# **interface [*id*].[*vlan id*]**

* Enter sub-interface configuration mode

*Sub-Interfaces are used in conjunction with vlans. To route vlans, a sub-interface must be created for each respective VLAN.*

Router(config-subif)# **encapsulation dot1Q [*vlan id*]**

* Configures the interface to use IEEE 802.1Q encapsulation on frames

*While 802.1Q is not the only networking standard supporting vlans, it is the most prominent. The vlan id should be a vlan existing on a connected switch desired to be routed.*

Router(config-subif)# **ip address [*ip*] [*mask*]**

* Configure an IP for the sub-interface, just like any normal interface

// TFTP Server Configuration

Router(config)# **tftp-server [*local file address*]**

* Host local files on the router

// Telephony Service Configuration

Router(config)# **telephony-service**

* Enter the CUCM interface

Router(config-telephony)# **load [*ip phone version*] [*firmware name*]**

* Updates the Cisco IOS Telephony Services configuration file for a specific type of IP phone

*Cisco IP phones update themselves with new phone firmware whenever they are initially started or reloaded. When a phone starts or reboots, the phone reads the configuration file to determine the name of the firmware file it should load and then looks for that firmware file on a TFTP server. The firmware name should be defined without a file extension.*

Router(config-telephony)# **max-ephones [number]**

* Define the maximum number of SCCP phones that can register to the CUCM

Router(config-telephony)# **max-dn [number]**

* Define the max number of voice ports that can open

Router(config-telephony)# **ip source-address [*ip*] port [*voice port*]**

* Define the interface on the router that will act as the call server

*The default voice port is 2000.*

Router(config-telephony)# **create cnf-files**

* Generate configuration files for SCCP phones

// Ephone and Dial Number Configuration

Router(config)# **ephone-dn [*number*]**

* Enter the ephone dial number interface

Router(config-ephone-dn)# **number [*phone number*]**

* Set the phone number of the ephone

Configurations

Cisco 2811 VoIP Router

Router#show running-config

version 12.4

service timestamps debug datetime msec

service timestamps log datetime msec

no service password-encryption

hostname Router

boot-start-marker

boot-end-marker

logging message-counter syslog

no aaa new-model

memory-size iomem 25

no network-clock-participate slot 1

dot11 syslog

ip source-route

ip cef

ip dhcp pool PC

network 10.0.10.0 255.255.255.0

default-router 10.0.10.254

ip dhcp pool VOIP

network 10.0.20.0 255.255.255.0

default-router 10.0.20.254

option 150 ip 10.0.20.254

no ipv6 cef

multilink bundle-name authenticated

voice-card 0

no dspfarm

voice-card 1

no dspfarm

vtp domain CCNP

vtp mode transparent

archive

log config

hidekeys

vlan 2,10,20

interface FastEthernet0/0

no ip address

shutdown

duplex auto

speed auto

interface FastEthernet0/1

no shutdown

no ip address

duplex auto

speed auto

h323-gateway voip interface

interface FastEthernet0/1.10

encapsulation dot1Q 10

ip address 10.0.10.254 255.255.255.0

interface FastEthernet0/1.20

encapsulation dot1Q 20

ip address 10.0.20.254 255.255.255.0

interface FastEthernet0/0/0

interface FastEthernet0/0/1

interface FastEthernet0/0/2

interface FastEthernet0/0/3

interface Serial0/1/0

no ip address

shutdown

interface Serial0/2/0

no ip address

shutdown

clock rate 2000000

interface Serial0/2/1

no ip address

shutdown

clock rate 2000000

interface Serial0/3/0

no ip address

shutdown

clock rate 2000000

interface Serial0/3/1

no ip address

shutdown

clock rate 2000000

interface Vlan1

no ip address

ip forward-protocol nd

no ip http server

no ip http secure-server

tftp-server P00308000500.sbn

tftp-server P00308000500.loads

tftp-server flash:P00308000500.bin alias P00308000500

tftp-server flash:P00308000500.sb2

control-plane

voice-port 1/0/0

voice-port 1/0/1

voice-port 1/0/2

voice-port 1/0/3

voice-port 1/1/0

voice-port 1/1/1

telephony-service

load 7960-7940 P00308000500

max-ephones 10

max-dn 10

ip source-address 10.0.20.254 port 2000

system message deez

max-conferences 5 gain -6

transfer-system full-consult

create cnf-files version-stamp Jan 01 2002 00:00:00

ephone-dn 1

number 911

ephone-dn 2

number 420

ephone 1

device-security-mode none

mac-address 001B.D512.B06E

button 1:1

ephone 2

device-security-mode none

mac-address 001B.D512.A57A

button 2:2

line con 0

line aux 0

line vty 0 4

login

scheduler allocate 20000 1000

end

PoE Switch

Switch#sh run

version 12.2

no service pad

service timestamps debug uptime

service timestamps log uptime

no service password-encryption

hostname Switch

boot-start-marker

boot-end-marker

no aaa new-model

system mtu routing 1500

vtp domain CCNP

vtp mode transparent

authentication mac-move permit

ip subnet-zero

crypto pki trustpoint TP-self-signed-1928519808

enrollment selfsigned

subject-name cn=IOS-Self-Signed-Certificate-1928519808

revocation-check none

rsakeypair TP-self-signed-1928519808

spanning-tree mode pvst

spanning-tree etherchannel guard misconfig

spanning-tree extend system-id

vlan internal allocation policy ascending

vlan 2

name forleft

vlan 3

name forright

vlan 4-5,7

vlan 10

name PC

vlan 12

vlan 20

name VOIP

vlan 30

name Expedia

vlan 40

name forty

vlan 99

vlan 100

name Microsoft

vlan 192

name Guest

vlan 996

name CUSTOMER\_NATIVE

interface FastEthernet0/1

switchport access vlan 10

switchport mode access

interface FastEthernet0/2

switchport access vlan 10

switchport mode access

interface FastEthernet0/3

switchport access vlan 10

switchport mode access

interface FastEthernet0/4

switchport access vlan 10

switchport mode access

interface FastEthernet0/5

switchport access vlan 10

switchport mode access

interface FastEthernet0/6

switchport access vlan 20

switchport mode access

spanning-tree portfast

interface FastEthernet0/7

switchport access vlan 20

switchport mode access

spanning-tree portfast

interface FastEthernet0/8

switchport access vlan 20

switchport mode access

spanning-tree portfast

interface FastEthernet0/9

switchport access vlan 20

switchport mode access

spanning-tree portfast

interface FastEthernet0/10

switchport access vlan 20

switchport mode access

spanning-tree portfast

interface FastEthernet0/11

switchport trunk encapsulation dot1q

switchport mode trunk

interface FastEthernet0/12

interface FastEthernet0/13

interface FastEthernet0/14

interface FastEthernet0/15

interface FastEthernet0/16

interface FastEthernet0/17

interface FastEthernet0/18

interface FastEthernet0/19

interface FastEthernet0/20

interface FastEthernet0/21

interface FastEthernet0/22

interface FastEthernet0/23

interface FastEthernet0/24

interface FastEthernet0/25

interface FastEthernet0/26

interface FastEthernet0/27

interface FastEthernet0/28

interface FastEthernet0/29

interface FastEthernet0/30

interface FastEthernet0/31

interface FastEthernet0/32

interface FastEthernet0/33

interface FastEthernet0/34

interface FastEthernet0/35

interface FastEthernet0/36

interface FastEthernet0/37

interface FastEthernet0/38

interface FastEthernet0/39

interface FastEthernet0/40

interface FastEthernet0/41

interface FastEthernet0/42

interface FastEthernet0/43

interface FastEthernet0/44

interface FastEthernet0/45

interface FastEthernet0/46

interface FastEthernet0/47

interface FastEthernet0/48

interface GigabitEthernet0/1

interface GigabitEthernet0/2

interface GigabitEthernet0/3

interface GigabitEthernet0/4

interface Vlan1

no ip address

shutdown

ip classless

ip http server

ip http secure-server

ip sla enable reaction-alerts

line con 0

line vty 0 4

login

line vty 5 15

login

end

Conclusion

Considering the amount of time that I personally spend using VoIP services, digging deeper has been an intriguing experience which I hope to learn more about. I can speak with friends in England from America in almost real time. It is fascinating how we are thousands of miles apart but can remain close due to a protocol first released before I was born. Now I got to investigate that protocol, likely one of the most meaningful ones in our world today.