Dial up server

Hey there! There is a newer version of this guide: Dial-up pool

Contents

Prerequisites

Hardware Requirements

- A hardware modem (not a software modem or "winmodem")
- A Linux device (e.g. x86 computer, Raspberry Pi, etc.) to communicate with the a modem as the dial-in server
- A client device with a modem (any type)
- Some form of telephony connection to link the two modems

The hardware I've chosen to use is:

- Generic x86_64 PC
- Matrix MX Modem (more on this later)
- USB to RS-232 serial adapter (DE-9) to connect to the modem (with hardware flow control)
- DE-9 to DB-25 serial adapter
- Linksys PAP2T analog telephone adapter (ATA)
- x86-based Windows 95 PC with a US Robotics Sportster 28800 ISA modem

Software Requirements

- Ubuntu server 18.04

- ppp
- getty
- Asterisk

Choosing Modem Hardware

Modem hardware varies greatly, but this project doesn't require anything beyond standard protocols between your ISP and client modems (V.22, V.32, etc.).
This tutorial *should* work with any dial-up modem that presents itself as a serial device to the operating system, including cheap USB modems, ISA modems, PCI modems and of course external RS-232 serial modems.

Note: You will have a lot of trouble using a softmodem/winmodem (https://en.wikipedia.org/wiki/Softmodem)! You are much better off using a hardware-based modem.

I'll be using an external serial modem + USB-to-RS-232 adapter for this tutorial. Using dedicated serial hardware has the advantage of being easy to troubleshoot and scaling up to dozens of lines (if you have enough desk space and USB ports).
My modem is an MX Modem made by MATRIX, but I was unable to find *any* information about it. Obviously the next step was to crack it open to figure out its capabilities.

Inside is a XECOM XE1414C 14.4 kbps modem in a single component package. The PDF manual is here (https://dogemicrosystems.ca/files/dial/XECOM_XE1414.pdf).
To connect the modem to the ISP-side computer, I'm using a generic USB to RS-232 adapter and a DE-9 to DB-25 adapter.

Matrix MX Modem with top cover removed

MX Modem and RS232 adapters

The Telephone Network

We need a way to connect our ISP modem to clients. There are many ways to approach this:

1. Use the actual PSTN (i.e. real phone lines)
2. Use a PBX to provide local connectivity
3. Build your own circuity (not covered here as it would require extra configuration)
4. Build a fake PSTN using VoIP ATAs and a software PBX

I've gone with the fourth option. Here's the breakdown:

1. Asterisk - a VoIP PBX - is configured on the dial-in server to accept connections from two SIP client accounts and route calls between them.

2. A Linksys PAP2T ATA - which supports two phone lines - is set up as both of those SIP clients connected to the PBX.
3. The ISP-side modem is connected to the first line, and the client device to the second line.

Asterisk Setup

1. Install asterisk

```
sudo apt-get install asterisk
```

2. Append configuration for the two SIP clients to the end of /etc/asterisk/sip.conf

```
[pap2t-ispmodem]
context=default
type=friend
secret=password
qualify=200                     ; Qualify peer is no more than 200ms away
host=dynamic                    ; This device registers with us
directmedia=yes                 ; Send RTP directly to the peer to reduce latency
and jitter
regexten=881
nat=no

[pap2t-client]
context=default
type=friend
secret=password
qualify=200                     ; Qualify peer is no more than 200ms away
host=dynamic                    ; This device registers with us
directmedia=yes                 ; Send RTP directly to the peer to reduce latency
and jitter
regexten=882
nat=no
```

3. Edit /etc/asterisk/extensions.conf and make two changes:
   Search for [default] (should be around line 672) and comment out
   include => demo
   Underneath that line, add a new line with
   exten => _X!,1,Dial(SIP/pap2t-ispmodem, 20)
   The _X! tells this dial plan rule to match any number a client dials and send the call to the pap2t-ispmodem client
4. Enable the asterisk service so it starts on boot
   sudo systemctl enable asterisk
5. Start Asterisk
   sudo systemctl start asterisk
6. Open the Asterisk console to confirm your ATA lines are registered
   sudo asterisk -rvvvv

```
dialupserver*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport
Comedia    ACL Port     Status      Description
pap2t-client/pap2t-client 10.1.8.112                               D  No
No             5061     OK (7 ms)
pap2t-ispmodem/pap2t-ispm 10.1.8.112                               D  No
```

```
No             5060     OK (6 ms)
2 sip peers [Monitored: 2 online, 0 offline Unmonitored: 0 online, 0 offline]
```

If you make changes to your configuration after starting Asterisk, you can use the reload command in the console to reload the configuration.

ATA SIP registration example:

```
    -- Registered SIP 'pap2t-ispmodem' at 10.1.8.112:5060
       > Saved useragent "Linksys/SPA2102-5.2.5" for peer pap2t-ispmodem
       NOTICE[4253]: chan_sip.c:24592 handle_response_peerpoke: Peer 'pap2t-ispmodem'
is now Reachable. (7ms / 200ms)
    -- Registered SIP 'pap2t-client' at 10.1.8.112:5061
       > Saved useragent "Linksys/SPA2102-5.2.5" for peer pap2t-client
       NOTICE[4253]: chan_sip.c:24592 handle_response_peerpoke: Peer 'pap2t-client'
is now Reachable. (5ms / 200ms)
```

Successful call example:

```
  == Using SIP RTP CoS mark 5
       > 0x7fa234059e30 -- Strict RTP learning after remote address set to:
10.1.8.112:16482
    -- Executing [88567682@default:1] Dial("SIP/pap2t-client-00000002", "SIP/pap2t-
ispmodem, 20") in new stack
  == Using SIP RTP CoS mark 5
    -- Called SIP/pap2t-ispmodem
    -- SIP/pap2t-ispmodem-00000003 is ringing
       > 0x7fa1f8252350 -- Strict RTP learning after remote address set to:
10.1.8.112:16384
    -- SIP/pap2t-ispmodem-00000003 answered SIP/pap2t-client-00000002
    -- Channel SIP/pap2t-ispmodem-00000003 joined 'simple_bridge' basic-bridge
<5f9983eb-3f93-49d1-90b3-c91c545c2d38>
    -- Channel SIP/pap2t-client-00000002 joined 'simple_bridge' basic-bridge
<5f9983eb-3f93-49d1-90b3-c91c545c2d38>
       > Bridge 5f9983eb-3f93-49d1-90b3-c91c545c2d38: switching from simple_bridge
technology to native_rtp
       > Remotely bridged 'SIP/pap2t-client-00000002' and 'SIP/pap2t-ispmodem-
00000003' - media will flow directly between them
       > 0x7fa1f8252350 -- Strict RTP learning after remote address set to:
10.1.8.112:16384
       > 0x7fa234059e30 -- Strict RTP learning after remote address set to:
10.1.8.112:16482
    -- Channel SIP/pap2t-client-00000002 left 'native_rtp' basic-bridge <5f9983eb-
3f93-49d1-90b3-c91c545c2d38>
    -- Channel SIP/pap2t-ispmodem-00000003 left 'native_rtp' basic-bridge <5f9983eb-
3f93-49d1-90b3-c91c545c2d38>
  == Spawn extension (default, 88567682, 1) exited non-zero on 'SIP/pap2t-client-
00000002'
       > 0x7fa1f8252350 -- Strict RTP learning after remote address set to:
10.1.8.112:16384
```

ATA Configuration

I won't go into much detail on ATA configuration since the topic has been beaten to death on various forums. This is the process to get basic communication:

1. Set up both lines on the ATA to register to the PBX with usernames 'pap2t-ispmodem' and 'pap2t-client' and the password 'password'
2. Use the G.711 μ-law codec
3. Disable every echo cancellation option in your ATA (see here for PAP2T instructions (https://www.voip-info.org/linksys-pap2t))
4. Set the jitter buffer to be as small as possible

Congratulations! You now have your own voice network.

The Dial-in Server

1. Install your Debian-based Linux distribution of choice (not covered here)
2. Update to latest packages and reboot if required

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

3. Connect the USB to RS-232 adapter and confirm it shows up as /dev/ttyUSBXXX (ls /dev/ to check). In my case, it presents as /dev/ttyUSB0
   My serial adapter is a "ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adaptor"
4. Install ppp (and getty if your distro doesn’t have it by default)

```
sudo apt-get install ppp mgetty
```

5. Many of the old guides were written when inittab was still around, but it's 2019 and systemd has taken over.
   We need to create a systemd service for mgetty, so edit /lib/systemd/system/mgetty.service with your text editor of choice as root.

```
[Unit]
Description=External Modem
Documentation=man:mgetty(8)
Requires=systemd-udev-settle.service
After=systemd-udev-settle.service

[Service]
Type=simple
ExecStart=/sbin/mgetty /dev/ttyUSB0
Restart=always
PIDFile=/var/run/mgetty.pid.ttyUSB0

[Install]
WantedBy=multi-user.target
```

6. Configure mgetty by editing /etc/mgetty/mgetty.config with your text editor of choice as root.
   Comment out everything except the debug level, and append the section for configuring the serial device:

```
debug 9

port ttyUSB0
 port-owner root
 port-group dialout
 port-mode 0660
 data-only yes
 ignore-carrier no
 toggle-dtr yes
 toggle-dtr-waittime 500
 rings 2
 #autobauding yes
 speed 9600
```

7. Enable the mgetty service so it starts on boot:
   sudo systemctl enable mgetty.service
8. Start mgetty:
   sudo systemctl start mgetty.service
9. Configure ppp by editing /etc/ppp/options
   Like above, comment out everything except these settings:

```
# Define the DNS server for the client to use
ms-dns 8.8.8.8
# async character map should be 0
asyncmap 0
# Require authentication
auth
# Use hardware flow control
crtscts
# We want exclusive access to the modem device
lock
# Show pap passwords in log files to help with debugging
show-password
# Require the client to authenticate with pap
+pap
# If you are having trouble with auth enable debugging
debug
# Heartbeat for control messages, used to determine if the client connection has
dropped
lcp-echo-interval 30
lcp-echo-failure 4
# Cache the client mac address in the arp system table
proxyarp
# Disable the IPXCP and IPX protocols.
noipx
```

10. Create a device option file by editing /etc/ppp/options.ttyUSB0

```
local
lock
nocrtscts
192.168.32.1:192.168.32.105
netmask 255.255.255.0
```

```
noauth
proxyarp
lcp-echo-failure 60
```

11. Create the user for PAP authentication:
    `sudo useradd -G dialout,dip,users -m -g users -s /usr/sbin/pppd dial`
12. Set a password:
    `sudo passwd dial`
13. Edit /etc/ppp/pap-secrets and append the username and password (same as you entered above, quotes included):
    `dial * "dial" *`
14. Enable packet forwarding for IP4 by editing /etc/sysctl.conf:
    `net.ipv4.ip_forward=1`
15.
16. The last step for the dial-up server is to configure the firewall to allow traffic forwarding from PPP out onto the network (and off to the Internet).
    1. On Linux distributions with iptables, you need to add a line to /etc/rc.local to enable masquerading. If your Ethernet interface is named eth0, you would add this line:
       `iptables -t nat -A POSTROUTING -s 192.168.32.0/24 -o eth0 -j MASQUERADE`
    2. On modern Ubuntu installs, ufw is used as a frontend to iptables, so the procedure is a bit different. Follow this guide (https://help.ubuntu.com/lts/serverguide/firewall.html.en#ip-masquerading), but you can omit -o eth0 and use -s 192.168.32.0/24.

Troubleshooting

When using an external modem, the choice of USB to RS-232 adapter seems to be crucial. There aren't many requirements, but you must use an adapter that supports hardware flow control. If you need to purchase an adapter, you can either get one that explicitly says it supports hardware flow control ($$$), or play the eBay lottery and buy a half-dozen different models and hope one of them works.

I ran into a bug in Debian 9.5 with my USB to serial adapter using the ch341 driver, where setting the baud rate was not working on some Linux kernels. (Seems to be this bug (https://bugzilla.redhat.com/show_bug.cgi?id=1235715))

To troubleshoot modem communication and baud rate settings, use minicom (or screen) to open a session over serial and try different settings (or read your modem's manual!). Sending the command 'AT" followed by a new line should result in your modem replying 'OK'.
If you're getting nothing at all out of your modem, perform a serial loopback test (http://www.ni.com/tutorial/3450/en/)

If mgetty is not answering incoming calls, it may be having trouble communicating with your modem. Check the logs in `/var/log/mgetty/` to determine the problem. You may need to set a modem initialization string in the mgetty device config file, so check your modem's manual for help on this.
