Class 2: more systemd, udev and device trees

ESE3005

a word on “listening sockets”

(from stackoverflow, courtesy of David M. Syzdek):

“A client socket does not listen for incoming connections, it initiates an outgoing connection to the server. The server socket listens for incoming connections.

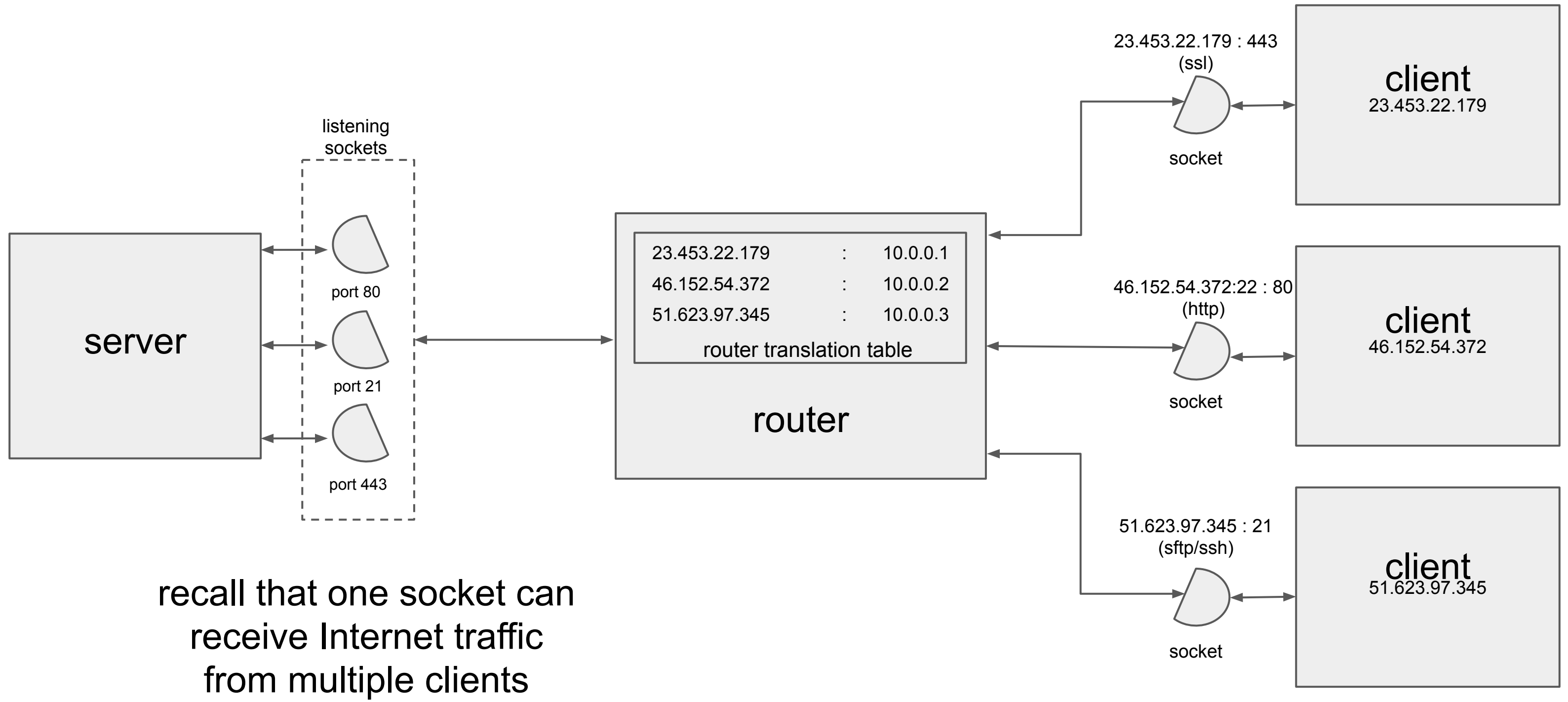
A server creates a socket, binds the socket to an IP address and port number (for TCP and UDP), and then listens for incoming connections. When a client connects to the server, a new socket is created for communication with the client (TCP only). A polling mechanism is used to determine if any activity has occurred on any of the open sockets.

A client creates a socket and connects to a remote IP address and port number (for TCP and UDP). A polling mechanism can be used (`select()`, `poll()`, `epoll()`, etc) to monitor the socket for information from the server without blocking the thread.

In the case that the client is behind a router which provides NAT (network address translation), the router re-writes the address of the client to match the router's public IP address. When the server responds, the router changes its public IP address back into the client's IP address. The router keeps a table of the active connections that it is translating so that it can map the server's responses to the correct client.”

there are a few things going on here

- first is the system-level operations of the network, amongst server, router, client (and the associated sockets)
- second are the Unix-like system calls made to kernel (on either the server, router or client machines), in order to accomplish various tasks
- various aspects of the TCP/IP or UDP protocols
- one of the chief initial operations of *systemd* is to create a number of *listening socket*s for all the services that require it; note that systemd was developed initially under RedHat Linux, which is used predominantly on server platforms; your Beaglebone, is unlikely to require many listening sockets, although it is important to keep in mind that the Beaglebone Black is configured to be a web server by default under Debian Linux from beagleboard.org



recall that one socket can
receive Internet traffic
from multiple clients

systemd system and start-up features

the systemd system and service manager provides the following main features:

- *Socket-based activation* — At boot time, systemd creates listening sockets for all system services that support this type of activation, and passes the sockets to these services as soon as they are started. This not only allows systemd to start services in parallel, but also makes it possible to restart a service without losing any message sent to it while it is unavailable: the corresponding socket remains accessible and all messages are queued.
  Systemd uses *socket units* for socket-based activation.
- *Bus-based activation* — System services that use D-Bus for inter-process communication can be started on-demand the first time a client application attempts to communicate with them. Systemd uses *D-Bus service files* for bus-based activation.
- *Device-based activation* — System services that support device-based activation can be started on-demand when a particular type of hardware is plugged in or becomes available. Systemd uses *device units* for device-based activation.
- *Path-based activation* — System services that support path-based activation can be started on-demand when a particular file or directory changes its state. Systemd uses *path units* for path-based activation.

systemd system and start-up features (cont'd)

- *Mount and automount point management* — Systemd monitors and manages mount and automount points. Systemd uses *mount units* for mount points and *automount units* for automount points.
- *Aggressive parallelization* — Because of the use of socket-based activation, systemd can start system services in parallel as soon as all listening sockets are in place. In combination with system services that support on-demand activation, parallel activation significantly reduces the time required to boot the system.
- *Transactional unit activation logic* — Before activating or deactivating a unit, systemd calculates its dependencies, creates a temporary transaction, and verifies that this transaction is consistent. If a transaction is inconsistent, systemd automatically attempts to correct it and remove non-essential jobs from it before reporting an error.
- *Backwards compatibility with SysV init* — Systemd supports SysV init scripts as described in the *Linux Standard Base Core Specification*, which eases the upgrade path to systemd service units.

. source: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-managing_services_with_systemd

systemd units

Systemd introduces the concept of *systemd units*. These units are represented by unit configuration files located in one of the directories listed in Table 10.2, “Systemd Unit Files Locations”, and encapsulate information about system services, listening sockets, and other objects that are relevant to the init system. For a complete list of available systemd unit types, see Table 10.1, “Available systemd Unit Types”.

Table 10.1. Available systemd Unit Types

Unit Type	File Extension	Description
Service unit	`.service`	A system service.
Target unit	`.target`	A group of systemd units.
Automount unit	`.automount`	A file system automount point.
Device unit	`.device`	A device file recognized by the kernel.
Mount unit	`.mount`	A file system mount point.
Path unit	`.path`	A file or directory in a file system.
Scope unit	`.scope`	An externally created process.
Slice unit	`.slice`	A group of hierarchically organized units that manage system processes.
Snapshot unit	`.snapshot`	A saved state of the systemd manager.
Socket unit	`.socket`	An inter-process communication socket.
Swap unit	`.swap`	A swap device or a swap file.
Timer unit	`.timer`	A systemd timer.

Table 10.2. Systemd Unit Files Locations

Directory	Description
`/usr/lib/systemd/system/`	Systemd unit files distributed with installed RPM packages.
`/run/systemd/system/`	Systemd unit files created at run time. This directory takes precedence over the directory with installed service unit files.
`/etc/systemd/system/`	Systemd unit files created by `systemctl enable` as well as unit files added for extending a service. This directory takes precedence over the directory with runtime unit files.

relationship with *SysV*

- under SysV, the system operates under various *runlevels*.
  - for example a runlevel of 0 corresponds to “halt the system (shutdown)”, a runlevel of 1 corresponds to *single-user mode*, while runlevels 2-5 correspond to *multi-user mode*

- with systemd, these runlevels have been replaced by *target units*; you can determine your boards current default target as follows:

```
$ systemctl get-default
```

TARGET NAMES	**SysV runlevel**	**description**
poweroff.target	0	halt the system
rescue.target	1, S	single-user mode; admin functions like checking filesystem
multi-user.target	2-4	regular multi-user mode with no windowing display
graphical.target	5	regular multi-user mode with windowing display
reboot.target	6	reboot the system
emergency.target		emergency shell on the main console

try this:

```
$ systemctl list-units --type=target
```

Creating your own *systemd* service

source: https://www.linode.com/docs/quick-answers/linux/start-service-at-boot/