📁 LPIC-2 / 🌐 Examen 205 - Configuración de Red

205.2 Configuración de red avanzada y resolución de problemas

Teoría

La administración de red en LPIC-2 requiere comprender y configurar tecnologías más allá de la simple asignación de direcciones IP. También implica dominar herramientas para diagnosticar problemas de red más allá de un simple `ping`.

Configuraciones de Red Avanzadas:

1. **Agregación de Enlaces (Bonding / Teaming / EtherChannel / LACP):**
    - **Propósito:** Combinar múltiples interfaces de red físicas (ej: `eth0`, `eth1`) en una única interfaz lógica (`bond0`).
    - **Beneficios:** Aumentar el ancho de banda agregado o proporcionar redundancia (si falla una interfaz física, el array sigue funcionando).
    - **Módulos del Kernel:** Requiere el módulo `bonding`.
    - **Modos de Bonding Comunes (`/proc/net/bonding/bondX` muestra el modo y estado):**
        - `mode=0` (balance-rr): Round-robin. Tráfico distribuido secuencialmente entre las interfaces. Mayor ancho de banda, pero puede causar desorden de paquetes.
        - `mode=1` (active-backup): Solo una interfaz está activa; si falla, otra toma el control. Proporciona redundancia/failover.
        - `mode=4` (802.3ad - LACP): Dynamic Link Aggregation Control Protocol. Requiere soporte en el switch. Negocia los parámetros de agregación dinámicamente.
    - **Configuración (Diferencias según el método):**
        - **Tradicional Debian (`/etc/network/interfaces`):** Define una interfaz `bond0` con `iface bond0 inet ...`, `bond-mode <modo>`, `bond-slaves <lista_interfaces>`. Las interfaces físicas (`eth0`, `eth1`) se configuran como `manual` y sin dirección IP.
        - **Tradicional Red Hat (`/etc/sysconfig/network-scripts/`):** Crea un archivo `ifcfg-bond0` (`DEVICE=bond0`, `BOOTPROTO=...`, `ONBOOT=yes`, `BONDING_OPTS="mode=<modo>"`). Cada interfaz miembro (`ifcfg-eth0`, `ifcfg-eth1`) tiene `MASTER=bond0` y `SLAVE=yes`.
        - **NetworkManager:** Se configura una conexión de tipo "bond" a través de `nmcli` (`nmcli connection add type bond con-name bond0 ifname bond0 mode <modo>`) y luego se crean conexiones "slave"

para cada interfaz física (`nmcli connection add type ethernet slave-type bond con-name eth0-slave ifname eth0 master bond0`). Gestiona archivos en `/etc/NetworkManager/system-connections/`.
    - **Netplan:** Se define un tipo `bond` en el archivo `.yaml` con sus interfaces miembros y modo.
    - **systemd-networkd:** Se definen archivos `.netdev` para el bond y archivos `.network` que asocian las interfaces físicas al bond.
2. **Puentes de Red (Bridging):**
    - **Propósito:** Conectar múltiples segmentos de red (interfaces físicas y/o virtuales) a nivel de Capa 2 (Enlace de Datos), funcionando como un switch. El tráfico se reenvía en base a direcciones MAC.
    - **Usos:** Principalmente para virtualización, conectando interfaces virtuales de VMs (ej: `vnet0`) a una interfaz de puente (`br0`) que a su vez está conectada a una interfaz física (`eth0`). Esto permite que las VMs aparezcan en la misma red que el host físico.
    - **Herramientas:** `bridge` (iproute2, moderna) o `brctl` (bridge-utils, antigua).
    - **Configuración (Diferencias según el método):** Similar a bonding, se configura en `/etc/network/interfaces`, `ifcfg-*`, NetworkManager, Netplan o systemd-networkd.
        - **Tradicional Debian:** Define una interfaz `br0` con `iface br0 inet ...`, `bridge_ports <lista_interfaces>`. Las interfaces físicas se configuran como `manual`.
        - **Tradicional Red Hat:** Crea un archivo `ifcfg-br0` (`TYPE=Bridge`, `BOOTPROTO=...`, `ONBOOT=yes`). Cada interfaz miembro tiene `TYPE=Ethernet`, `BRIDGE=br0`.
        - **NetworkManager/Netplan/systemd-networkd:** Definen conexiones/unidades de tipo bridge y asocian interfaces como puertos del bridge.
3. **VLANs (Virtual Local Area Networks):**
    - **Propósito:** Segmentar lógicamente una red física en múltiples redes virtuales. Permite agrupar dispositivos (independientemente de su ubicación física) en la misma red virtual.
    - **Funcionamiento:** Se utiliza el estándar 802.1q para "etiquetar" (tagging) las tramas Ethernet con un ID de VLAN. Interfaces en la misma VLAN pueden comunicarse; interfaces en VLANs diferentes requieren un router o switch de Capa 3.
    - **Configuración:** Se crean subinterfaces virtuales en una interfaz física o de bond (ej: `eth0.10` para la VLAN ID 10 en `eth0`). Estas subinterfaces reciben sus propias direcciones IP.
    - **Módulo del Kernel:** Requiere el módulo `8021q`.

    - **Herramientas:** `vconfig` (bridge-utils, antigua) o `ip link add link <interfaz> name <nombre> type vlan id <id>` (iproute2, moderna).
    - **Configuración Persistente (Diferencias):** Se configura en los archivos/herramientas de red persistentes de la distribución/método, definiendo la subinterfaz VLAN y su configuración IP.

Herramientas Avanzadas de Resolución de Problemas:

- **`tcpdump` / `wireshark`:** Permiten capturar y analizar paquetes de red que pasan por una interfaz. Esencial para ver el tráfico real y diagnosticar problemas de protocolo, comunicación, etc. `tcpdump` es de línea de comandos; Wireshark es una GUI.
    - `tcpdump -i <interfaz>`: Captura tráfico en una interfaz.
    - `-n`: No resolver IPs a nombres de host.
    - `-nn`: No resolver IPs ni puertos a nombres.
    - `-v`, `-vv`, `-vvv`: Aumentar verbosidad.
    - `-S`: Mostrar números de secuencia absolutos.
    - `-A`: Mostrar paquetes en ASCII.
    - `-w <archivo>`: Escribir paquetes a un archivo para análisis posterior (ej: con Wireshark).
    - Filtros: Puedes filtrar por host, puerto, protocolo (ej: `tcpdump -i eth0 host 192.168.1.10 and port 22`).
- **`ip neigh` / `arp`:** Muestran la caché ARP (Address Resolution Protocol), que mapea direcciones IP a direcciones MAC en la red local. Útil para verificar si puedes resolver la MAC de un host local dado su IP. `ip neigh show` (moderno), `arp -a` (antiguo).
- **`netcat` (`nc`):** (Visto en 109.3) Sigue siendo útil para probar si un puerto TCP/UDP está abierto en un host remoto.
- **`ss` / `netstat`:** (Visto en 109.3) Para ver conexiones activas y puertos a la escucha, útil para verificar si un servicio está realmente esperando conexiones en la red.

Metodología de Resolución de Problemas (Escenarios Complejos):

- **Divide y Vencerás:** Empieza por los problemas más básicos (Capa 1: ¿está enchufado? Capa 2: ¿resuelve ARP? Capa 3: ¿`ping` a la IP? Capa 4: ¿`nc` al puerto?) antes de pasar a capas superiores (DNS, HTTP, SSH).
- **Aísla el Problema:** ¿Falla la conexión a un solo host o a todos? ¿Falla un solo servicio o todos? ¿Falla solo desde esta máquina o desde otras también? ¿Falla en una dirección o en ambas?
- **Usa `tcpdump`:** Cuando los problemas básicos no revelan la causa, captura tráfico para ver qué paquetes se envían, reciben, o si hay errores de protocolo.
- **Verifica Firewalls:** Recuerda revisar los firewalls locales (`firewalld`, `ufw`, `iptables`) y de red.
- **Revisa Logs:** Logs del sistema (auth.log, secure, messages, journalctl) y logs de la aplicación o servicio de red.