Nützliches / Tests

- Layer2 Paket senden
- N2N Neigboarhood

Layer2 Paket senden

Beschreibung:

Kleines python script mit dem man ein layer2 Datenpaket senden kann.

Bedienung:

```
sendframe -h
usage: sendframe.py [-h] -d DEVICE -m {senden,empfangen}



-d device = eth0 eth1 enp6s0 n2n0 etc
-m senden oder empfangen

Beispiel Ziel Computer im empfangsmodus zuerst starten :
./sendframe.py -d n2n0 -m empfangen
Dann auf dem Quellcomputer paket senden
./sendframe.py -d n2n0 -m senden
```

Das Script auch zum sendframe.py

```
#!/usr/bin/python3

import socket
import sys
import argparse
import signal



def signal_handler(sig, frame):
    print('Programm wurde mit STRG+C beendet.')
    sys.exit(0)
```

```
def send_frame(iface):
    # Erstellt einen RAW-Socket
    s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW)
    # Bindet den Socket an die angegebene Schnittstelle
    s.bind((iface, 0))

    # Beispiel eines Ethernet-Frames mit einer leeren Nutzlast
    # Ethernet Header: Ziel MAC, Quelle MAC, Ethertype (0x0800 für IPv4)
    dst_mac = b'\xff\xff\xff\xff\xff\xff'  # Broadcast MAC-Adresse
    src_mac = s.getsockname()[4]
    ethertype = b'\x08\x00'
    payload = b'Hello, network!'
    frame = dst_mac + src_mac + ethertype + payload

    # Sendet den Frame
    s.send(frame)
    print(f'Paket gesendet auf {iface}')

def receive_frame(iface):
    # Erstellt einen RAW-Socket
    s = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0003))
    # Bindet den Socket an die angegebene Schnittstelle
    s.bind((iface, 0))

    print(f'Warte auf Pakete auf {iface}...')
    while True:
        # Empfängt den Frame
        frame, addr = s.recvfrom(65535)
        src_mac = frame[6:12]
        print(f'Paket empfangen von MAC: {":".join(format(x, "02x") for x in src_mac)}')

def main():
    parser = argparse.ArgumentParser(description='Einfaches Skript zum Senden und Empfangen von Ethernet-
Frames')
    parser.add_argument('-d', '--device', required=True, help='Netzwerkgerät (z.B. eth0)')
    parser.add_argument('-m', '--mode', required=True, choices=['senden', 'empfangen'], help='Modus: senden
oder empfangen')

    args = parser.parse_args()
```

```
    # Registriert das STRG+C Interrupt-Signal
    signal.signal(signal.SIGINT, signal_handler)

    if args.mode == 'senden':
        send_frame(args.device)
    elif args.mode == 'empfangen':
        receive_frame(args.device)

if __name__ == '__main__':
    main()
```

N2N Neigboarhood

Beschreibung:

Das Programm LLDPd (Link Layer Discovery Protocol daemon) ist ein Open-Source-Daemon, der das Link Layer Discovery Protocol (LLDP) implementiert. LLDP ist ein standardisiertes Netzwerkprotokoll, das von Netzwerkgeräten verwendet wird, um ihre Identität, Fähigkeiten und Nachbarschaftsinformationen auf dem Netzwerk weiterzugeben. Hier sind die Hauptfunktionen und Merkmale von LLDPd:

1. **Geräteidentifikation**: LLDPd ermöglicht es Netzwerkgeräten, Informationen wie den Gerätenamen, die Portbezeichnung und die Systembeschreibung an andere Geräte im Netzwerk weiterzugeben. Dies erleichtert die Verwaltung und das Troubleshooting von Netzwerken.
2. **Topologieerkennung**: Mit LLDPd können Netzwerkadministratoren eine detaillierte Übersicht über die Netzwerkstruktur erhalten. Geräte, die LLDP unterstützen, senden regelmäßig LLDP-Informationen aus, die von anderen Geräten empfangen und angezeigt werden können.
3. **Kapazitätenaustausch**: Geräte können über LLDPd Informationen über ihre Fähigkeiten und Ressourcen austauschen, wie z.B. die unterstützten Geschwindigkeiten, die Duplex-Konfiguration und andere technische Spezifikationen.
4. **Energieverwaltung**: LLDPd unterstützt auch LLDP-MED (Media Endpoint Discovery), eine Erweiterung von LLDP, die zusätzliche Funktionen wie die Verwaltung von Energieeinstellungen und die Priorisierung von Netzwerkverkehr für Voice-over-IP (VoIP) bietet.
5. **Integration mit Netzwerkmanagement-Tools**: LLDPd kann in verschiedene Netzwerkmanagement-Tools integriert werden, um eine zentrale Verwaltung und Überwachung von Netzwerken zu ermöglichen.
6. **Konfigurationsdateien und Plugins**: LLDPd verwendet konfigurierbare Dateien und unterstützt Plugins, die zusätzliche Funktionen hinzufügen oder das Verhalten des Daemons anpassen können.
7. **Multiplattformunterstützung**: LLDPd ist für verschiedene Betriebssysteme verfügbar, einschließlich Linux, FreeBSD und anderen Unix-ähnlichen Systemen.

Zusammengefasst dient LLDPd dazu, die Verwaltung und Überwachung von Netzwerken zu erleichtern, indem es detaillierte Informationen über die Netzwerkgeräte und ihre Verbindungen bereitstellt.

Wir nutzen es um zu schauen ob der n2n Tunnel funkltioniert

Installation:

```
apt-get install lldpd

#Dieses dann an der Brücke anwenden
lldpd -I br0

#Nun die Ausgabe mit:
lldpcli show neighbors
```

Ausgabe (Dieser Befehl wurde auf n2n Node A ausgefüjhrt und zeigt den neigbour n2n node B im lokalen netz:

```
-------------------------------------------------------------------------------
LLDP neighbors:
-------------------------------------------------------------------------------
Interface:    n2n0, via: LLDP, RID: 2, Time: 12 days, 14:36:25
  Chassis:
    ChassisID:    mac 26:da:23:37:41:42
    SysName:      pubxxxxxxx.local
    SysDescr:     Debian GNU/Linux 11 (bullseye) Linux 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12)
x86_64
    MgmtIP:       192.168.178.xxx
    MgmtIface:    3
    MgmtIP:       fd00::949e:xxxx:fe27:xxxx
    MgmtIface:    3
    Capability:   Bridge, on
    Capability:   Router, off
    Capability:   Wlan, off
    Capability:   Station, off
  Port:
    PortID:       mac 4a:30:8d:xx:xx:xx
    PortDescr:    n2n0
    TTL:          120
-------------------------------------------------------------------------------
```