MKC-NBS: Úkol č. 2

Kryptografie

Filip Paul

04.03.2022

Rámec Ethernet II:

Zadání:

Níže jsou uvedeny bajty dvou rámců typu Ethernet II, tak jak byly zachyceny programem Wireshark. Tento program ve svých výpisech neuvádí návěští rámců ani jejich kontrolní součet, takže tato pole v zadání nehledejte.

1. rámec:

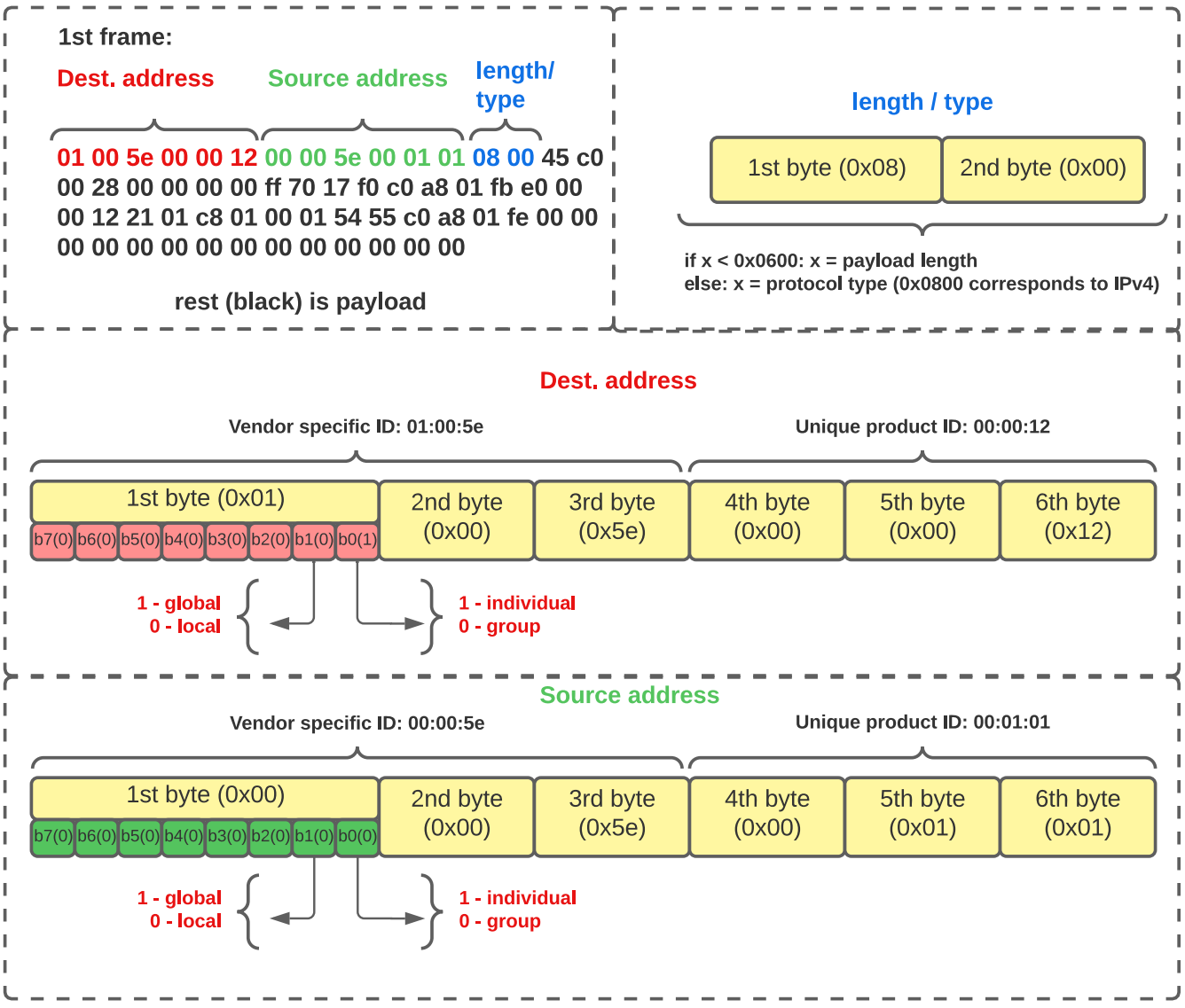
01 00 5e 00 00 12 00 00 5e 00 01 01 08 00 45 c0
00 28 00 00 00 00 ff 70 17 f0 c0 a8 01 fb e0 00
00 12 21 01 c8 01 00 01 54 55 c0 a8 01 fe 00 00
00 00 00 00 00 00 00 00 00 00 00 00

2. rámec:

ff ff ff ff ff ff f0 f3 36 af f4 54 08 06 00 01
08 00 06 04 00 01 00 07 0d af f4 54 18 a6 ac 01
00 00 00 00 00 00 18 a6 ad 9f 06 01 04 00 00 00
00 02 01 00 03 02 00 00 05 01 03 01

Z uvedených údajů zjistěte pro oba rámce cílovou a zdrojovou adresu a zjištěné adresy charakterizujte (globální - skupinová - individuální, globální správa adresy - lokální správa adresy, případně uveďte výrobce karty). Dále určete protokol, jehož zpráva je v těle daného rámce přenášena. Potřebné informace lze získat například na následujících stránkách: iana.org a adminsub.net.

Vypracování:



Protokol STP:

Zadání:

Je dána ethernetová síť sestávající ze čtyř přepínačů s identifikátory 1, 2, 3 a 4. Tyto přepínače jsou propojeny podle níže uvedeného obrázku. Pro zadanou síť určete pomocí algoritmu STP její kostru. Přenesené zprávy zapište do přehledné tabulky a zjištěnou kostru zakreslete s vyznačením, které porty jsou zapnuty a které blokovány.

Vypracování:

PYTHON CBC.py

Přílohy:

EthernetFrame.py

```python
from frames import first_frame, second_frame
frames = [first_frame, second_frame]

def parseMAC(MAC_frame):
    MAC_str = ""
    for byte,i in zip(MAC_frame,range(len(MAC_frame))):
        if i == 0:
            individual_group = byte & 1
            global_local = (byte & 0b10) >> 1
            if individual_group == 1:
                individual_group = "group"
            elif individual_group == 0:
                individual_group = "individual"
            if global_local == 1:
                global_local = "local"
            elif global_local == 0:
                global_local = "global"
            print(f"first byte: 0x{byte:02X} -> 0b{bin(byte)[2:].zfill(8)} ->{individual_group} and {global_local} adress.\nMAC: ",end = "")

        MAC_str += f"{byte:02X}:"
    print(f"{MAC_str[:-1]} -> Vendor specific part: {MAC_str[0:8]}")

for fr, i in zip(frames, range(len(frames))):

    print(f"\n\nFRAME {i+1}:")
    dest_addr = fr[0:6]
    src_addr = fr[6:6+6]
    length_type = fr[12: 12+2]
    payload = fr[14:]

    print("Destination address:")
    parseMAC(dest_addr)

    print("\nSource address:")
    parseMAC(src_addr)

    print(f"\nlengt/type: 0x{length_type[0]:02X}{length_type[1]:02X} -> DEC: ", end = '')
    len_type_value = int(f"{length_type[0]:02X}{length_type[1]:02X}",16)
    if len_type_value < 1536:
        print(f"Value is smaller than DEC: 1536 -> lenght of payload is: {len_type_value} bytes")
    else:
        #Common eth_type values:
        if len_type_value == 0x800:
            eth_type = "IPv4"
        elif len_type_value == 0x86DD:
            eth_type = "IPv6"
        elif len_type_value == 0x806:
            eth_type = "ARP"
        elif len_type_value == 0x811:
            eth_type = "VLAN"

        print(f"Value is bigger than DEC: 1536 -> Ethertype is: {eth_type}")
```