Laborator APC - 6

Rețele locale cu comutatoare Ethernet

Obiective

În această lucrare de laborator vom studia arhitectura și funcționarea rețelelor locale bazate pe comutatoare Ethernet (punți transparente): echipamente, configurare, algoritmi și protocoale. În acest scop, vom examina comunicațiile dintre echipamente și starea lor pe parcursul unor scenarii de utilizare care scot în evidență aspecte esențiale ale funcționării lor.

Vom studia trei teme principale:

- Dirijarea cadrelor MAC (Ethernet) în rețele locale realizate cu punți transparente.
- Reducerea unei topologii fizice de tip graf (cu legături redundante, pentru asigurarea rezilienței) la o topologie logică de tip arbore de acoperire (cu conectivitate completă, dar fără bucle) folosind Spanning Tree Protocol (STP).
- Crearea mai multor rețele locale distincte, numite LAN-uri virtuale (VLAN), folosind o infrastructură fizică comună, alcătuită din comutatoare Ethernet interconectate.

Protocoalele utilizate în aceste rețele au fost standardizate de IEEE și sunt descrise în standardele IEEE 802.1D - MAC Bridges (2004) și IEEE 802.1Q - Bridges and Bridged Networks (2014).

Precondiții

Pentru a putea efectua experimentele și a interpreta rezultatele trebuie să studiați în prealabil capitolele din materialul de curs (și eventual bibliografia suplimentară) care prezintă noțiunile de bază privind rețelele locale cu punți transparente.

Software și echipamente

Experimentele vor folosi implementări software ale echipamentelor și protocoalelor, disponibile în sistemul de operare Linux (în special, pachetul `bridge-utils`). Veți captura și analiza comunicațiile dintre echipamente folosind analizoarele de protocoale `tcpdump` și `wireshark`.

Fiecare student (sau echipă de 2 studenți) va lucra pe un calculator care rulează sistemul de operare Linux. Rețeaua studiată este emulată pe fiecare calculator folosind platforma de emulare `netkit`. Fiecare echipament din rețea este implementat ca o mașină virtuală Linux și este accesibil prin intermediul unui terminal (pentru configurare, examinarea stării, executarea unor utilitare, etc.).

A. Dirijarea cadrelor în rețele locale cu punți transparente

În prima parte a lucrării vom studia algoritmul folosit de punți pentru dirijarea cadrelor MAC.

Punțile dirijează un cadru către destinație folosind adresa MAC destinație din antetul cadrului și un tabel care asociază fiecărei adrese MAC (cunoscute) portul de ieșire prin care trebuie transmis cadrul pentru a ajunge la adresa respectivă.

Punțile transparente au fost proiectate ca să permită comunicații unicast și broadcast sau multicast între toate echipamentele pe care le interconectează fără o configurare prealabilă ("plug-and-play"). Prin urmare, punțile trebuie să descopere singure fiecare echipament conectat la rețea, precum și

portul de ieşire prin care trebuie să retransmită un cadru pentru a ajunge la destinație. Această informație este obținută din cadrele recepționatee: adresa sursă din antetul cadrului este asociată portului prin care a fost recepționat cadrul (mai multe detalii în materialul de curs).

Dirijarea cadrelor în rețele cu punți este specificată în standardul IEEE 802.1D (2004).

Studiu de caz

Experimentele vor fi efectuate în rețeaua locală prezentată în Figura 1. Rețeaua este alcătuită din 4 calculatoare cu interfețe Ethernet interconectate prin 2 punți (bridge/switch) Ethernet. Mai precis, c1 și c2 sunt conectate direct la comutatorul s1, iar c3 și c4 sunt conectate la un hub Ethernet (emulat de `netkit`), care este conectat la portul `eth1` al comutatorului s2.

Pentru fiecare interfață, Figura 1 indică numele său (de exemplu, `eth0`) și adresa MAC (cei 2 octeți mai puțin semnificativi, ceilalți 4 octeți fiind 0). În plus, interfețele calculatoarelor au alocate și adrese IP (de exemplu, 1.0.0.1).

Proiectul `netkit` cu care începeți această primă parte a lucrarii conține toate echipamentele din Figura 1, cu interfețele deja configurate. Pentru început, va trebui să activați și să configurați funcția de punte transparentă pe mașinile virtuale s1 și s2.

Apoi, veți efectua o serie de experimente simple pentru a observa funcționarea algoritmului de dirijare folosit de punți: veți genera trafic între calculatoare folosind comanda `ping`, apoi veți examina și analiza traficul, operațiile efectuate de punțile s1 și s2 și modificarea stării punților.

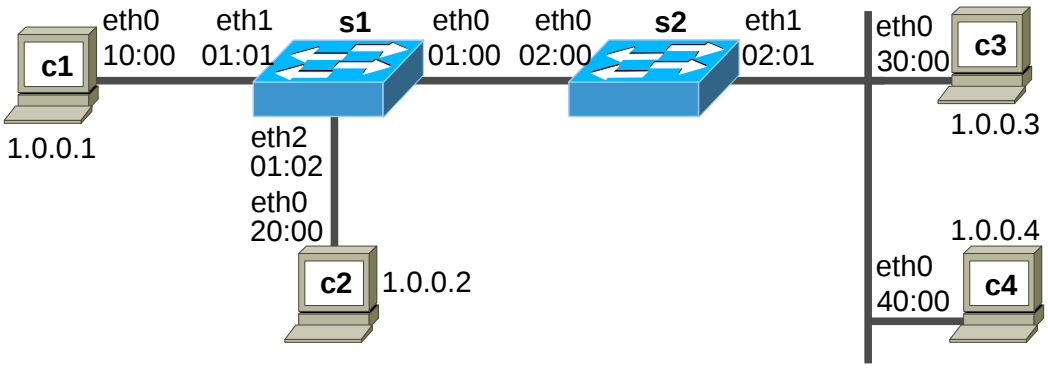

Figura 1: Rețeaua utilizată pentru studiul dirijării cadrelor în rețele cu punți.

A.1. Inițializarea rețelei

A.1.1. Pentru a porni emularea rețelei din Figura 1, executați comanda `lstart` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

A.1.2. Examinați configurarea inițială a echipamentelor folosind comenzile următoare:

`ifconfig`	afișează starea interfețelor
`brctl show`	afișează rezumatul stării punților

Sunt activate toate interfețele utilizate în rețea? Sunt configurate corect adresele MAC și IP? Este activată funcția de punte (bridge) pe s1 și s2?

A.2. Inițializarea punților

Veți crea și configura punțile pe mașinile virtuale s1 și s2 folosind comanda `brctl` (bridge control). Pentru a vizualiza un rezumat al funcționalității oferite de această comandă, executați în terminalul unei mașini virtuale comanda: `man brctl`.

A.2.1. Configurați punțile s1 și s2. Pentru s1 folosiți comenzile următoare (similar pentru s2):

`brctl addbr br0`	crează (adaugă) puntea `br0`
`brctl addif br0 eth0`	atașează interfața `eth0` la puntea `br0`
`brctl addif br0 eth1`	atașează interfața `eth1` la puntea `br0`
`ifconfig br0 up`	activează puntea `br0`

În general, ar trebui activat protocolul STP pe fiecare punte cu comanda `brctl stp br0 on`. STP nu este însă necesar în această topologie și, pentru a simplifica analiza traficului, nu îl mai activăm.

A.2.2. Examinați starea punților s1 și s2 folosind comenzile următoare:

`brctl show`	afișează rezumatul stării punților
`brctl showstp br0`	afișează starea interfețelor punții `br0` conform STP (pentru moment, ne interesează doar starea fiecărui port: `disabled`, `forwarding`, etc.)
`brctl showmacs br0`	afișează tabelul de adrese al punții `br0`

Este activată puntea pe s1 și s2? Sunt atașate la fiecare punte toate interfețele indicate în Figura 1? Sunt toate porturile în starea `forwarding`?

A.3. Dirijarea cadrelor: comunicație de la c3 la c4

A.3.1. Capturați traficul pe interfața `eth1` a comutatoarelor s1 și s2 folosind comenzile:

```
s1:# tcpdump -i eth1 -s 0 -w /hostlab/s1e1tst1.cap
```

```
s2:# tcpdump -i eth1 -s 0 -w /hostlab/s2e1tst1.cap
```

A.3.2. Inițiați o comunicație între c3 și c4 folosind comanda `ping`:

```
c3:# ping -c 3 1.0.0.4
```

Există conectivitate între c3 și c4? De ce la primul dialog ICMP Echo Request/Reply întârzierea este mult mai mare decât la celelalte?

A.3.3. Opriți `tcpdump` (`Ctrl-C`) și analizați ce s-a întâmplat în timpul experimentului:

- Afișați tabelele de adrese al punților s1 și s2 folosind comanda `brctl`.
- Vizualizați traficul capturat folosind `wireshark`, executând într-un terminal al calculatorului gazdă comenzile:

  ```
  wireshark -r s1e1tst1.cap &
  ```

  ```
  wireshark -r s2e1tst1.cap &
  ```

- Examinați traficul capturat și explicați pentru fiecare pachet operațiile efectuate de fiecare punte și modificările din tabelul său de adrese.

A.3.4. O înregistrare din tabelul de adrese al unei punți expiră (este ștearsă automat din tabel) dacă sursa cu adresa respectivă nu mai transmite timp de 300 secunde (durată configurabilă). Explicați de

ce este necesar acest lucru. Verificați expirarea înregistrărilor pentru experimentele efectuate.

A.4. Dirijarea cadrelor: comunicație de la c1 la c4

Repetați punctele A.3.1-A.3.4 pentru o comunicație de la c1 la c4.

A.5. Terminarea primei părți a lucrării

Pentru a termina emularea rețelei din Figura 1 executați comanda `lcrash` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

B. Protocolul STP

Algoritmul de dirijare studiat în prima parte a lucrării nu poate să funcționeze corect decât într-o topologie de rețea fără bucle. Pe de altă parte, în practică, este posibil ca această condiție să nu fie îndeplinită, fie pentru că dorim o topologie cu legături redundante, pentru a asigura reziliență, fie din cauza unei interconectări eronate.

Din acest motiv, în proiectarea acestui tip de rețea locală a fost prevăzut ca punțile să poată reduce (automat) o topologie fizică de tip graf (cu bucle) la o topologie logică de tip arbore de acoperire (cu conectivitatea completă, dar fără bucle) folosind Spanning Tree Protocol (STP). Pentru a elimina buclele, punțile blochează o parte dintre porturi. astfel încât să elimine din topologie legăturile care închid bucle (păstrând însă conectivitate completă).

Pe de altă parte, protocolul STP asigură și reconfigurarea rețelei în urma defectării unei legături active. În acest caz, în esență, punțile descoperă și crează un alt arbore de acoperire, reactivând o parte din porturile blocate anterior.

Protocolul STP este specificat în standardul IEEE 802.1D (2004).

Studiu de caz

Experimentele vor fi efectuate în rețeaua locală prezentată în Figura 2. Rețeaua este alcătuită din 2 calculatoare cu interfețe Ethernet interconectate prin 4 punți (bridge/switch) Ethernet. Observați că în topologia acestei rețele există mai multe bucle (s2-s4, s1-s2-s4-s3, etc.).

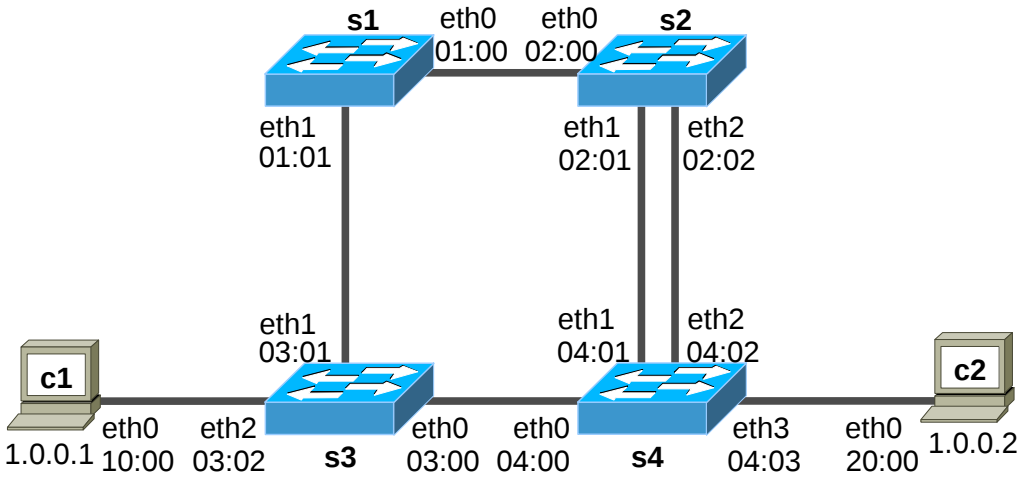

Figura 2: Rețeaua utilizată pentru experimente cu protocolul STP.

Proiectul `netkit` cu care începeți a doua parte a lucrarii conține echipamentele din Figura 2 cu interfețele și punțile configurate complet. În această rețea a fost activat pe fiecare punte și protocolul STP, cu comanda `brctl stp br0 on`. După inițializarea rețelei, veți efectua câteva experimente pentru a observa cum funcționează protocolul STP, care este arborele de acoperire creat inițial și cum este reconfigurat în cazul unei defecțiuni.

B.1. Inițializarea rețelei

B.1.1. Începeți emularea rețelei din Figura 2: executați comanda `lstart` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

B.1.2. Examinați configurarea inițială a echipamentelor folosind comenzile următoare:

`ifconfig`	afișează starea interfețelor
`brctl show`	afișează rezumatul stării punților

Sunt activate toate interfețele utilizate în rețea? Sunt configurate corect adresele MAC și IP? Este activată funcția de punte (bridge) pe mașinile virtuale s1-s4? Sunt atașate la punți toate interfețele conform Figurii 2?

B.2. Arborele de acoperire inițial

B.2.1. Examinați starea punților și identificați arborele de acoperire inițial:

`brctl showstp br0`	afișează starea interfețelor punții `br0` conform STP.

Starea fiecărui port și arborele inițial sunt ilustrate în Figura 3, folosind notațiile următoare: RB = Root Bridge; RP = Root Port; DP = Designated Port; BP = Blocked Port. Interpretați informația afișată de comanda `brctl showstp` pentru fiecare punte și verificați că punțile au creat efectiv arborele din figură.

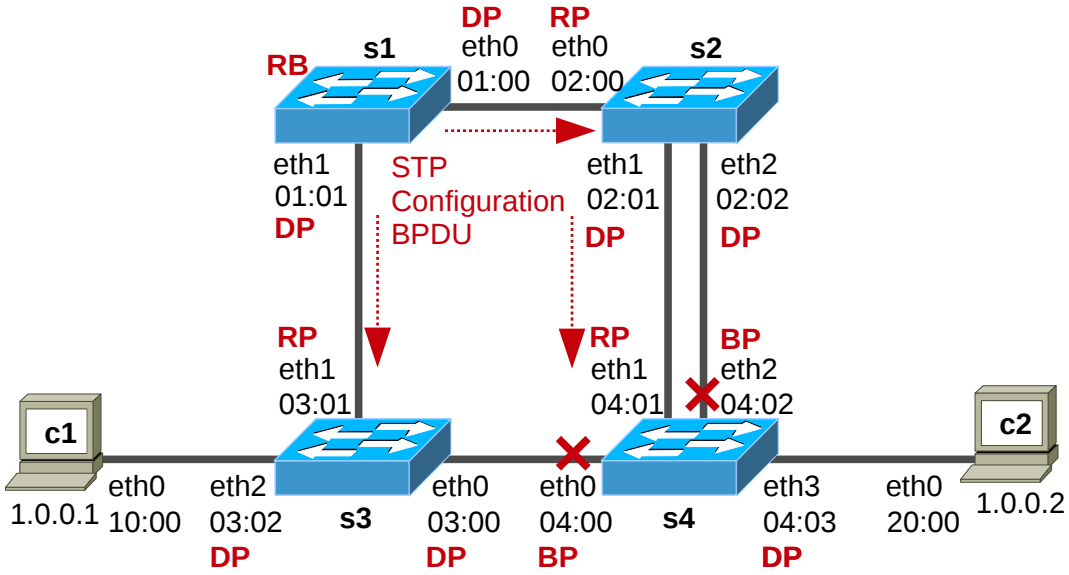

Figura 3: Arborele de acoperire inițial.

B.2.2. Capturați și examinați cadrele STP transmise de punți.

Puntea aleasă ca rădăcină a arborelui transmite periodic cadre STP Configuration BPDU (Bridge Protocol Data Unit). Celelalte punți retransmit aceste cadre pe porturile DP astfel încât sunt propagate în toată rețeaua (perioada este de 2 secunde, configurabilă). Astfel, punțile pot să verifice

că topologia de arbore este funcțională și să detecteze o eventuală pierdere a conectivității în urma unei defecțiuni (o interfață, o legătură sau o punte defectă).

Capturați cadrele transmise și recepționate de puntea s1 pe interfața `eth0` și de puntea s2 pe interfața `eth1` folosind comenzile următoare:

```
s1:# tcpdump -i eth0 -s 0 -nev
```

```
s2:# tcpdump -i eth1 -s 0 -nev
```

Analizați cadrele capturate și identificați diferențele dintre cele transmise de s1 și cele transmise de s2. Repetați pentru alte punți și porturi (de exemplu, s2/`eth2`).

Indicații: Adresa MAC destinație a cadrele STP (BPDU) este 01:80:C2:00:00:00. Aceasta este o adresă multicast rezervată pentru STP.

B.3. Dirijarea pachetelor pe arborele de acoperire

B.3.1. Inițiați captura pachetelor ARP și IP transmise și recepționate de puntea s3 pe interfața `eth1` și de puntea s4 pe interfața `eth0`, folosind comenzile:

```
s3:# tcpdump -i eth1 -s 0 -nev arp or ip
```

```
s4:# tcpdump -i eth0 -s 0 -nev arp or ip
```

B.3.2. Inițiați o comunicație între c1 și c2 folosind comanda `ping`:

```
c1:# ping -c 3 1.0.0.2
```

B.3.3. Opriți captura de pachete și apoi afișați tabelul de adrese al fiecărei punți folosind comanda:

```
brctl showmacs br0
```

B.3.4. Analizați traficul capturat și tabelele de adrese ale punților. Identificați porturile pe care au fost transmise pachetele ARP și ICMP în urma executării comenzii `ping`. Explicați modificările care apar în tabelele de adrese ale punților (aceste tabele conțin inițial doar adrese MAC ale punților).

B.4. Reconfigurarea arborelui în urma defectării unui port

B.4.1. Inițiați captura cadrelor transmise și recepționate de puntea s1 pe interfața `eth1`, puntea s2 pe interfața `eth1` și de puntea s4 pe interfața `eth0`:

```
s1:# tcpdump -i eth1 -s 0 -w /hostlab/s1e1stp1.cap
```

```
s2:# tcpdump -i eth1 -s 0 -w /hostlab/s2e1stp1.cap
```

```
s4:# tcpdump -i eth2 -s 0 -w /hostlab/s4e0stp1.cap
```

B.4.2. Inițiați o comunicație între c1 și c2 folosind `ping` (ICMP Request la interval de 1 secundă):

```
c1:# ping -i 1 1.0.0.2
```

B.4.3. Simulați defectarea interfeței `eth0` a switchului s2 executând comanda:

```
s2:# ifconfig eth0 down
```

B.4.4. Observați (din informația afișată de comanda `ping`) că după defectarea interfeței comunicația dintre c1 și c2 se întrerupe. Așteptați până când comunicația este reluată, apoi terminați execuția comenzii ping și captura pachetelor (`Ctrl-C`).

B.4.5. Afișați starea fiecărei punți cu comanda `brctl showstp br0.` Determinați arborele de acoperire creat de STP după defectarea interfeței pe baza informației afișate, verificați arborele pe

topologia rețelei și comparați-l cu arborele inițial (Figura 3).

B.4.6. Vizualizați traficul capturat folosind `wireshark`, executând într-un terminal al calculatorului gazdă comenzile:

```
wireshark -r s1e1stp1.cap &
wireshark -r s2e1stp1.cap &
wireshark -r s4e0stp1.cap &
```

Examinați traficul capturat și explicați ce se întâmplă în timpul reconfigurării arborelui. Ce cadre BPDU începe să transmită s2 după defectarea interfeței? Ce semnificație au cadrele BPDU TCN transmise de s4 și de s3? Cum reacționează s3 și s1 în urma primirii cadrelor BPDU TCN? Cum se modifică dirijarea cadrelor ICMP în timpul reconfigurării? Cât timp este întreruptă comunicația?

B.5. Terminarea celei de a doua părți a lucrării

Pentru a termina emularea rețelei din Figura 2, executați comanda `lcrash` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

C. Rețele locale virtuale (VLAN)

În practică, rețeaua unei organizații este divizată de obicei în mai multe subrețele interconectate prin rutere. Această divizare este motivată de rațiuni de securitate, performanță, fiabilitate și eventual administrative. De exemplu, se poate realiza câte o subrețea pentru fiecare departament, iar traficul dintre subrețele poate fi restricționat (prin configurarea ruterelor) în funcție de cerințele de securitate sau administrative (de pildă, accesul la resursele dintr-o subrețea este permis doar utilizatorilor din departamentul respectiv). În plus, se obține o partiționare a rețelei în domenii de broadcast separate, ceea ce permite îmbunătățirea performanței, scalabilității și fiabilității.

O soluție evidentă ar fi să construim separat fiecare subrețea folosind comutatoare Ethernet și apoi să interconectăm subrețelele prin rutere. Există însă o soluție mult mai flexibilă și mai eficientă: să construim o infrastructură comună pentru toate subrețelele, folosind comutatoare Ethernet, și să apoi să configurăm aceste comutatoarele pentru crea câte o rețea locală virtuală (VLAN) pentru fiecare subrețea. Tehnicile și protocoalele utilizate în acest scop sunt specificate în standardul IEEE 802.1.Q - Bridges and Bridged Networks (2014). În a treia (și ultima) parte a lucrării de laborator vom studia această soluție standard.

Studiu de caz

Experimentele vor fi efectuate în rețeaua prezentată în Figura 4: vom folosi o infrastructură alcătuită din comutatoarele s1 și s2 pentru a crea două LAN-uri virtuale, numite `vlan1` (roșu) și `vlan2` (verde) care corespund subrețelelor IP cu prefixele 1.0.1.0/24 și 1.0.2.0/24. Aceste subrețele IP vor fi interconectate prin ruterul r1.

Proiectul `netkit` inițial conține echipamentele din Figura 4 cu interfețele deja configurate. Trebuie să configurați comutatoarele și ruterul pentru a obține cele două VLAN-uri și a le interconecta.

C.1. Inițializarea rețelei

C.1.1. Începeți emularea rețelei din Figura 4, prin executarea comenzii `lstart` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

C.1.2. Examinați configurarea inițială a echipamentelor folosind comenzile următoare:

`ifconfig`	afișează starea interfețelor

`route`	afișează tabelele de rutare (pentru c1, c2 și r1)
`brctl show`	afișează rezumatul stării punților

Sunt activate toate interfețele utilizate în rețea? Sunt configurate corect adresele MAC și IP?

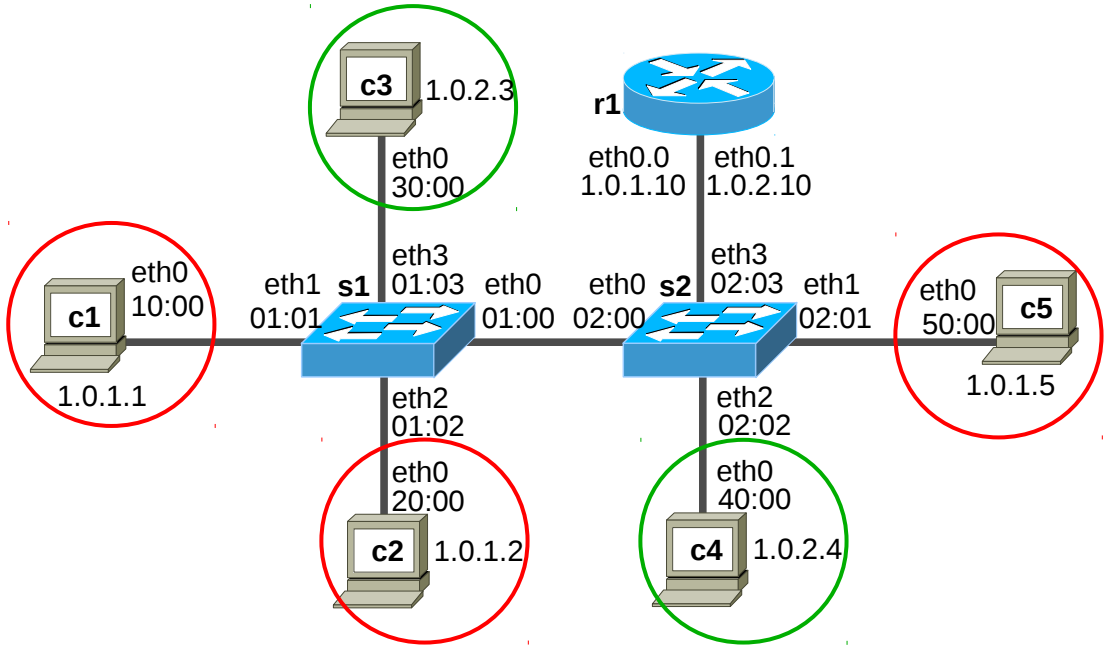

Figura 4: Rețea locală cu 2 VLAN-uri interconectate printr-un ruter.

C.2. Configurarea VLAN-urilor

Calculatoarele din `vlan1` sunt indicate printr-un cerc roșu, iar cele din `vlan2` printr-un cerc verde.

C.2.1. Pentru început, vom crea pe s1 două punți distincte, numite `vlan1` și `vlan2`, folosind comanda `brctl`. Pentru s1 puteți folosi comenzile următoare (similar pentru s2):

`brctl addbr vlan1`
`brctl addif vlan1 eth1`
`brctl addif vlan1 eth2`
`ifconfig vlan1 up`
`brctl addbr vlan2`
`brctl addif vlan2 eth3`
`ifconfig vlan2 up`
	crează puntea `vlan1`
	atașează `eth1,eth2` la `vlan1`
	
	activează `vlan1`
	similar pentru `vlan2`

Aceste comenzi ar fi suficiente pentru o rețea cu un singur comutator Ethernet. În exemplul nostru, VLAN-urile se extind pe 2 comutatoare interconectate. Pentru a configura legătura dintre s1 și s2 trebuie adăugate pe s1 comenzile următoare (similar pentru s2):

`vconfig add eth0 1`
`vconfig add eth0 2`
`brctl addif vlan1 eth0.1`
`brctl addif vlan2 eth0.2`
`ifconfig eth0.1 up`
`ifconfig eth0.2 up`
	crează interfața virtuală `eth0.1`
	crează interfața virtuală `eth0.2`
	adaugă `eth0.1` la `vlan1`
	adaugă `eth0.2` la `vlan2`
	activează interfețele virtuale

În urma executării acestor comenzi, pe legătura dintre s1 și s2 antetele cadrelor Ethernet sunt extinse cu un antet 802.1Q care specifică identificatorul (eticheta) VLAN-ului (VID) căruia îi

aparține cadrul. În exemplul nostru, conform comenzilor listate mai sus, VID = 1 pentru `vlan1` și VID = 2 pentru `vlan2`.

C.2.2. Examinați starea punților s1 și s2 folosind comenzile următoare:

```
brctl show
brctl showstp vlan1
brctl showstp vlan2
```

Sunt activate punțile pe s1 și s2? Sunt atașate la fiecare punte toate interfețele indicate în Figura 4? Sunt toate porturile în starea `forwarding`?

C.3. Comunicații între calculatoare din același VLAN

C.3.1. Inițiați captura cadrelor transferate pe legătura dintre c1 și s1 și pe legătura dintre s1 și s2:

```
s1:# tcpdump -i eth1 -s 0 -w /hostlab/s1e1vtst1.cap
s2:# tcpdump -i eth0 -s 0 -w /hostlab/s2e0vtst1.cap
```

C.3.2. Inițiați o comunicație între c1 și c5 folosind `ping`:

```
c1:# ping -c 3 1.0.1.5
```

C.3.3. Inițiați o comunicație între c3 și c4 folosind `ping`:

```
c3:# ping -c 3 1.0.2.4
```

C.3.4. Calculatoarele din VLAN1 nu pot să comunice cu cele din VLAN2. Pentru a verifia acest lucru, modificați temporar adresa IP a lui c3, astfel încât c1 și c3 să fie în aceeași subrețea, apoi testați dacă pot să comunice folosind `ping`:

```
c3:~# ifconfig eth0 1.0.1.3 netmask 255.255.255.0
c3:~# ping -c 2 1.0.1.1
PING 1.0.1.1 (1.0.1.1) 56(84) bytes of data.
From 1.0.1.3 icmp_seq=1 Destination Host Unreachable
From 1.0.1.3 icmp_seq=2 Destination Host Unreachable
...
c3:~# ifconfig eth0 1.0.2.3 netmask 255.255.255.0
```

C.3.5. Terminați captura și vizualizați traficul folosind `wireshark`, executând într-un terminal al calculatorului gazdă comenzile:

```
wireshark -r s1e1vtst1.cap &
wireshark -r s2e0vtst1.cap &
```

Examinați încapsularea cadrelor pe cele două legături. Explicați ce rol are transmiterea VID-ului în antetul cadrului.

C.4. Interconectarea VLAN-urilor

Pentru a interconecta cele două VLAN-uri trebuie să configurați ruterul r1 și să adăugați rutele statice implicite pe cele 5 calculatoare.

C.4.1. Configurați ruterul r1 cu comenzile următoare:

`ifconfig eth0 up`	activează interfața `eth0`
`vconfig add eth0 1`	crează interfețele virtuale

`vconfig add eth0 2`
`ifconfig eth0.1 1.0.1.10 netmask 255.255.255.0 up`
`ifconfig eth0.2 1.0.2.10 netmask 255.255.255.0 up`
	`eth0.1` și `eth0.2`
	alocă adrese IP interfețelor
	virtuale și le activează

Adăugați pe comutatorul s2 comenzile următoare, pentru a configura legătura dintre s2 și r1:

`vconfig add eth3 1`	crează interfața virtuală `eth3.1`
`vconfig add eth3 2`	crează interfața virtuală `eth3.2`
`brctl addif vlan1 eth3.1`	adaugă `eth3.1` la `vlan1`
`brctl addif vlan2 eth3.2`	adaugă `eth3.2` la `vlan2`
`ifconfig eth3.1 up`	activează interfețele virtuale
`ifconfig eth3.2 up`	

C.4.2. Configurați rutele implicite pe calculatoare. Exemple (similar pentru celelalte):

```
c1:~# route add default gw 1.0.1.10
```

```
c3:~# route add default gw 1.0.2.10
```

C.4.3. Examinați interfețele ruterului cu comanda `ifconfig` și apoi examinați tabelele de rutare ale ruterului și calulatoarelor cu comanda `route`.

Sunt activate și configurate corect interfețele virtuale? Există în tabelul de rutare rutele necesare pentru comunicații între cele două subrețele?

C.5. Comunicații între calculatoare din VLAN-uri diferite

C.3.1. Inițiați captura cadrelor transferate pe legătura dintre s1 și s2 și pe legătura dintre s2 și r1:

```
s1:# tcpdump -i eth0 -s 0 -w /hostlab/s1e0vtst2.cap
```

```
s2:# tcpdump -i eth3 -s 0 -w /hostlab/s2e3vtst2.cap
```

C.3.2. Inițiați o comunicație între c1 și c3 folosind `ping`:

```
c1:# ping -c 3 1.0.2.3
```

C.3.3. Terminați captura și vizualizați traficul folosind `wireshark`, executând într-un terminal al calculatorului gazdă comenzile:

```
wireshark -r s1e0vtst2.cap &
```

```
wireshark -r s2e3vtst2.cap &
```

Examinați încapsularea cadrelor pe cele două legături. Explicați ce rol are transmiterea VID-ului în antetul cadrului în acest caz. Identificați calea pe care sunt transmise pachetele și explicați.

C.6. Terminarea lucrării

Pentru a termina emularea rețelei din Figura 4, executați comanda `lcrash` într-un terminal al calculatorului gazdă, în directorul în care se află fișierele de configurare `netkit` ale acestei rețele.

Bibliografie

[1] O. Catrina. APC - Note de curs.

[2] Linux Foundation Wiki. Bridge. https://wiki.linuxfoundation.org/networking/bridge

[3] Uwe Böhme. Linux BRIDGE-STP-HOWTO.

http://www.tldp.org/HOWTO/BRIDGE-STP-HOWTO/index.html