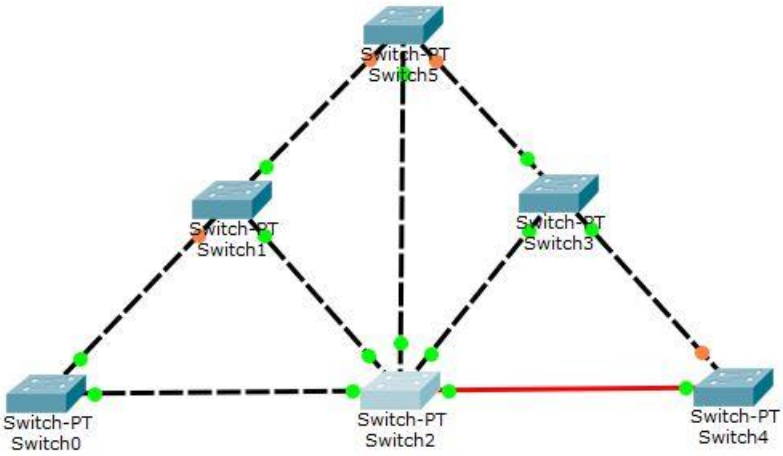
Structuri de date

Tema 3

Deadline: 15.05.2016

Sîrbu Lavinia Ștefania
sirbu.lavinia.stefania@gmail.com
Oprea George Bogdan
oprea.bg@gmail.com



Facultatea de Automatică și Calculatoare
Universitatea Politehnica din București
Anul universitar 2015-2016

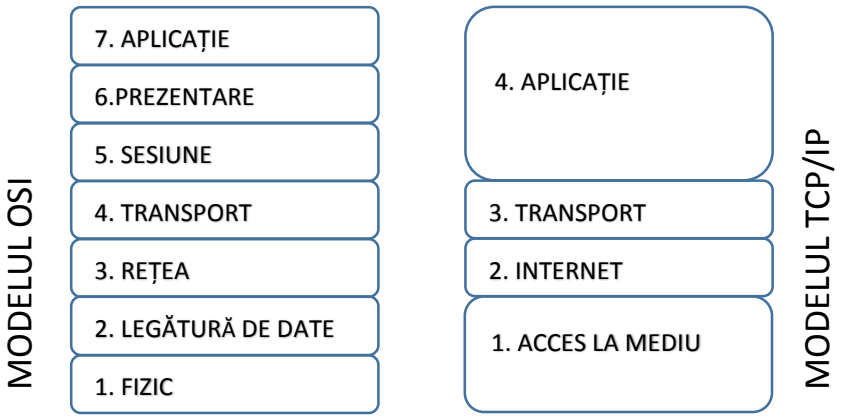
1 Obiective

În urma rezolvării acestei teme, studentul va fi capabil:

- să implementeze și să utilizeze grafuri
- să înțeleagă modul de lucru al Spanning Tree Protocol
- să implementeze o problemă de rețelistică și să înțeleagă problema care a dus la apariția STP

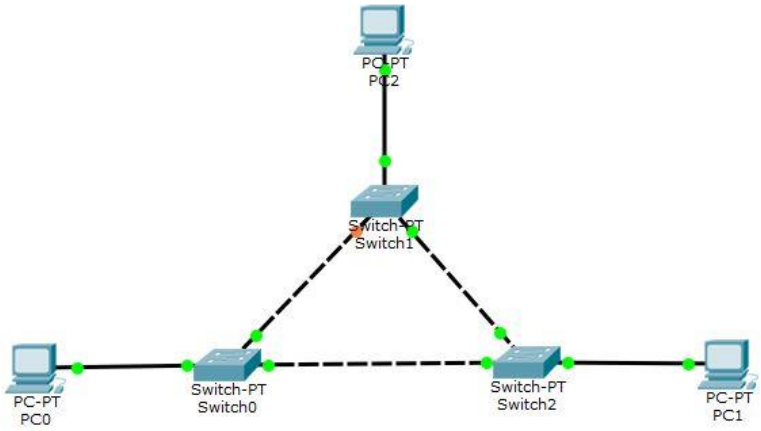
2 Descriere

La baza lucrului în rețea stau stivele de protocoale menite să reducă din complexitatea proiectării, astfel încât fiecare nivel să ofere anumite servicii nivelului superior, fără a îi spune însă și modul în care aceste servicii sunt implementate. Două arhitecturi de rețea importante sunt modelul de referință OSI(Open Systems Interconnection) și modelul TCP/IP(Transmission Control Protocol/Internet Protocol):



De interes pentru această temă este nivelul legătură de date din cadrul OSI, care asigură o formă de adresare a stațiilor în cadrul rețelei (adrese MAC) și oferă nivelurilor superioare acces la mediu. O problemă care apare însă la acest nivel este menținerea redundanței de la nivelul fizic, necesară pentru a evita problemele apărute atunci când un echipament de nivel 2 se defectează. Redundanța la nivel fizic implică interconectarea prin mai multe trasee a unor switch-uri (echipamente specifice nivelului 2) pentru a asigura că, în momentul în care unul se defectează, cadrele cu informații pot fi trimise pe o altă cale. Din punct de vedere logic însă, această redundanță aduce probleme, pentru că se creează bucle care vor duce la transmiterea anumitor cadre la infinit. Din acest motiv, asupra switch-urilor se aplică protocolul Spanning Tree Protocol, prin care vom ajunge de la un graf la un arbore de acoperire, astfel încât buclele să devină irelevante din punct de vedere logic.

Spre exemplu, în imaginea următoare, dacă legătura dintre Switch0 și Switch1 se defectează, PC0 și PC2 tot vor putea comunica prin legătura Switch0 – Switch2 – Switch1. Însă dacă nu am aplica STP și PC0 transmite un mesaj de BROADCAST (adresat tuturor stațiilor din rețea), el va rămâne la infinit în bucla Switch0 – Switch1 – Switch2 (pentru că un switch transmite mai departe pe toate porturile sale un mesaj de BROADCAST). Acest lucru nu este admisibil.



3 Spanning Tree Protocol

Scopul temei este implementarea unei variante simplificate a acestui protocol, în limbajul **C**. Pentru aceasta este necesar să introducem câteva noțiuni teoretice.

3.1 Costuri STP

Legătura dintre două switch-uri poate varia în funcție de viteza interfeței fizice (10Mb/s, 100Mb/s etc). Întrucât acest parametru este important, se definește un cost al unei legături astfel:

Viteză	**cost**
10 Mbps	100
100Mbps	19
1Gbps	4
10Gbps	1

3.2 Bridge ID

Un switch în cadrul protocolului STP este caracterizat de un BridgeID. Acesta este o valoare formată prin concatenarea priorității și a MAC-ului echipamentului. Prioritatea este cuprinsă între 4096 și 65535 (prin incrementări cu 4096) și, cu cât este mai mică, cu atât șansele ca un switch să devină root-bridge (switch-ul central) sunt mai mari. Spre exemplu, un switch cu prioritatea 32768 și MAC-ul AA-CD-BE-44-32-11 va avea BridgeID:

32768:AA-CD-BE-44-32-11

3.3 Alegerea root-bridge

Cunoscând BridgeID-urile tuturor switch-urilor, se alege root-bridge-ul ca fiind acela cu BridgeID minim. Se compară mai întâi prioritățile, iar dacă există mai multe switch-uri care au prioritatea minimă se vor compara MAC-urile (despre care știm cu siguranță că sunt unice, deci nu putem avea din nou situație de egalitate). De exemplu:

Sw2: 32768:AA-CD-BE-66-32-11

Sw1: 32768:AA-DD-BE-44-32-11 **=> *root-bridge = Sw2***

Sw3: 65535:AA-CD-BE-14-32-11

3.4 Alegerea root-ports

Pentru un switch, root-portul reprezintă acel port(ieșire) prin care va avea cea mai scurtă cale către root-bridge. Pentru alegerea root-portului, un switch va executa în ordine următoarele operații:

1. Costul cel mai mic raportat până la root-bridge(pe baza costurilor prezentate la punctul 3.1). Practic, fiecare switch, mai puțin root-bridge, se va uita la distanța până la root-bridge prin toate porturile sale și alege de aici doar drumurile de lungime minimă. Dacă distanța minimă se realizează printr-un singur port, atunci acela va deveni root-port.
2. Dacă există mai multe porturi prin care lungimea către root-bridge este minimă, root-port va fi portul conectat la switch-ul cu cel mai mic BridgeID.

3.5 Alegerea designated-ports

În primul rând, dacă un capăt al legăturii dintre 2 switch-uri este root-port, atunci celălalt switch va avea portul asociat respectivei legături ca designated-port.

Pentru toate celelalte porturi rămase libere, decizia alegerii designated-port pe o muchie se ia în felul următor:

1. Costul cel mai mic raportat până la root-bridge.
2. Portul conectat la switch-ul cu cel mai mic BridgeID.

3.6 Alegerea blocked-ports

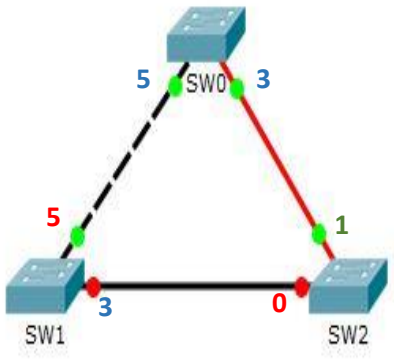
Toate muchiile pe care avem stabilit un designated port vor avea portul celălalt marcat la blocked-port(dacă acesta nu este deja root-port).

Așadar, spunem că pe o muchie se acceptă doar următoarele configurații ale porturilor root-port(RP) : designated- port(DP) sau designated-port(DP) : blocked port(BP).

Ca un rezumat, afirmăm faptul că toate porturile root-bridge-ului sunt designated-port; un switch poate avea un singur root-port și acela marchează prin ce port se ajunge cel mai repede către root-bridge; legături de forma RP : BP, RP : RP, DP : DP, BP : BP nu sunt permise.

Pentru a exemplifica punctele 3.4 – 3.6, folosim figura următoare (ignorăm culorile bulinelor din dreptul switch-urilor):

Considerăm că numerele de lângă switch-uri reprezintă numerele corespunzătoare porturilor. De asemenea, linia roșie e o legătură 10Mbs, cea neagră 100Mbs și cea punctată 1Gbs. Presupunem de asemenea că în urma BridgeID-urilor s-a decis că SW0 este root-bridge.



1. Cele două porturi ale SW0 sunt obligatoriu **DP**.
2. Conform tipurilor de legături, avem următoarele distanțe minime: SW1 - SW0 = 4 și SW2 – SW 0 = 23 (prin SW1).
3. Ca urmare a distanțelor minime, decidem că avem următoarele **RP**: SW1, port 5; SW2, port 0
4. Pentru că pe legătura SW1 – SW2 avem SW2, port 0 de tip **RP**, înseamnă că SW1, port 3 este **DP**.
5. Pe legătura SW2 – SW0 avem SW0 root-bridge. Cum SW2 are deja un **RP** si SW0, port 3 este **DP**, rezultă că SW2, port 1 trebuie să fie **BP**.

3.7 Criterii de reconvergență

Așa cum am spus, rolul STP-ului este eliminarea ciclurilor produse de redundanța adusă de nivelul fizic. În cazul în care o legătură se întrerupe, pot avea loc următoarele scenarii pentru ca întregul sistem să reconveargă la o configurație stabilă și corectă:

- Portul era în stare **blocked**, caz în care nu se produce nicio modificare.
- Portul era în stare **designated**, de asemenea nu se produce nicio modificare.
- Portul era în stare **root port**, caz în care switch-ul verifică dacă există o legătură alternativă cu root-bridge pe care să o treacă din blocked în root (nu este cazul în tema aceasta).
- Dacă nu se găsește o alternativă către root-bridge, se reface întreaga topologie prin STP (acest pas se va aplica în tema aceasta oricând au loc modificări de topologie).

Atenție! Pierderea unei legături implică verificarea criteriile de reconvergență pe ambele switch-uri implicate.

4 Cerințe

În vederea rezolvării fiecărei cerințe în parte, se vor citi comenzile din fișierul ***tasks.in*** (detalii referitoare la fișierul de intrare găsiți la secțiunea **5.1**).

4.1 Cerința 1

La primirea comenzii **c1**, se va afișa numărul switch-ului care a fost ales ca root-bridge și apoi se va afișa '\n'.

4.2 Cerința 2

La primirea comenzii **c2-1** se vor afișa, în ordinea crescătoare a numărului switch-ului, toate porturile care sunt root-ports. Ieșirea va respecta următorul format:

RP: număr_switch1(port_switch1) număr_switch2(port_switch2) ...

La primirea comenzii **c2-2** se vor afișa, în ordinea crescătoare a numărului switch-ului, toate porturile care sunt blocked-ports. Ieșirea va respecta următorul format:

BP: număr_switch1(port_switch1) număr_switch2(port_switch2) ...

După fiecare din cele 2 linii se va afișa '\n'.

La primirea comenzii **c2-3** se va afișa întreaga topologie în urma aplicării algoritmului STP, sub forma unei matrice. Considerând matricea de topologie A, aceasta va avea A[i][j] = 1 dacă există o legătură între switch-ul i și switch-ul j și portul corespunzător acestei muchii este RP pe switch-ul i. În rest valorile vor fi 0. Valorile se vor afișa separate printr-un singur spațiu, iar la sfârșitul fiecărei linii se va afișa doar '\n'(nu și spațiu după ultima valoare).

4.3 Cerința 3

La primirea comenzii **c3 număr_switch1 număr_switch2** se vor afișa, în ordine, switch-urile prin care trece un mesaj pentru a ajunge de la număr_switch1 la număr_switch2. Ieșirea va respecta următorul format:

număr_switch1 [număr_switch_X număr_switch_Y ...] număr_switch2

cu mențiunea că, dacă switch-urile 1 și 2 sunt direct legate atunci, evident, vor fi afișate doar ele la ieșire. De asemenea, switch-urile intermediare nu vor fi afișate între [și]. Cele două paranteze arată doar ca acele switch-uri pot lipsi în unele situații.

4.4 Cerința 4

La primirea comenzii **c4 număr_switch1 număr_switch2** se consideră că legătura dintre cele două switch-uri nu mai există. Ca urmare a acestei acțiuni, va trebui să verificați criteriile de reconvergență și să recalculați STP.

Atenție! Cerințele sunt strâns legate între ele în mod progresiv. Recomandăm parcurgerea în ordine a acestora. Deși punctajele obținute pe teste sunt independente unul față de celălalt, probabil nu veți obține punctajul pe o cerință dacă toate cele anterioare nu au fost rezolvate corect.

4.5 Bonus

În lumea reală putem avea switch-uri care se află în rețele diferite. Cu alte cuvinte, vom avea grupuri de switch-uri fără a exista legături prin care să ajungem de la un grup la altul. Aplicarea protocolului STP se va face asupra fiecărui grup în parte, mai puțin asupra acelor grupuri în care există un singur switch. În sensul rezolvării bonusului se vor face următoarele modificări la task-uri (rezolvarea bonusului nu trebuie să afecteze buna funcționare a cerințelor obligatorii):

- **c1**: se va face afișarea tuturor root-bridge-urilor posibile, în ordinea crescătoare a numărului switch-ului care este root-bridge.
- **c2**: se va face afișarea tuturor root-porturilor (**c2-1**)/blocked-porturilor (**c2-2**)/topologiilor grupurilor(**c2-3**), în ordinea crescătoare a numărului switch-ului care este root-bridge al.fiecărui grup.
- Funcționarea tuturor celorlalte comenzi va fi identică. Doar că se va lua în considerare faptul că switch-urile sunt împărțite în grupuri care nu comunică unul cu celălalt. O temă realizată corect însă nu ar trebui să aibă restul task-urilor influențate de faptul că acum sunt mai multe grupuri.

Se garantează faptul că, în cadrul unei rețele, numerele asociate switch-urilor sunt numere consecutive.

Atenție! Testarea bonusului se va face prin aceleași comenzi ca și pentru task-urile obligatorii. Va fi diferit doar conținutul fișierului ***topology.in*** *(detalii la secțiunea 5.1).*

5 Date de intrare-ieșire

5.1 Formatul fișierului de intrare

Vor fi primite ca parametri patru fișiere de intrare: ***initialise.in, topology.in, tasks.in, stp.out*** (ele vor fi parametri, așa că nu folosiți aceste nume în implementare. Sunt aici doar pentru a ușura explicațiile)**.**

În fișierul ***initialise.in*** se găsesc informații referitoare la fiecare switch în parte. Pe prima linie se află un număr întreg *n* care specifică numărul de switch-uri din configurație. Pe următoarele *n* linii se vor afla date despre fiecare switch în parte, astfel:

număr_switch prioritate_switch MAC_switch

având următoarele restricții:

0 <= număr_switch < n

4096 <= prioritate_switch <= 65535

MAC_switch este dat in hexazecimal și este de forma XX-XX-XX-XX-XX-XX

În fișierul ***topology.in*** se găsesc informații referitoare la muchiile din graf. Nu se cunoaște cu exactitate numărul de linii care se află în fișier. Pe o linie din acest fișier se găsesc următoarele:

număr_switch1 număr_switch2 port_swicth1 port_swicth2 tip_legătură

având următoarele restricții:

tip_legătură ϵ {10, 100, 1000, 10000}, reprezentând în ordine

legături de 10Mbs, 100Mbs, 1Gbs, 10Gbs

În fișierul **tasks.in** se găsesc cerințele efective. Acestea sunt câte una pe linie. Cerințele trebuie rezolvate în ordinea în care se află în fișier. Formatul cerințelor este cel specificat la punctul 4.

5.2 Formatul fișierului de ieșire

Afișarea se va face în fișierul **stp.out** (al patrulea argument din linia de comandă). Formatul pentru afișare este cel specificat la fiecare cerință în parte. La sfârșitul fiecărei linii afișate se va afișa '\n'. De asemenea aveți grijă la faptul că nu trebuie să aveți spații goale la sfârșitul liniilor. Dacă aveți întrebări referitoare la formatul de ieșire vă rugăm să le adresați pe forum pentru clarificare. Verificarea cu fișierele de referință se va face prin ***diff***, așa că până și un singur spațiu va influența negativ rezultatul unui test.

6 Punctaje

Cerința	**Punctaj**
Cerința 1	10
Cerința 2	30
Cerința 3	30
Cerința 4	20
Coding style, README, warning-uri	10
Bonus	20

7 Restricții și precizări

Temele trebuie să fie încărcate pe vmchecker. **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.

O rezolvare constă într-o arhivă de tip **zip** care conține toate fișierele sursă alături de un **Makefile**, ce va fi folosit pentru compilare, și un fișier **README**, în care se vor preciza detaliile de implementare. Toate fișierele se vor afla în rădăcina arhivei.

Makefile-ul trebuie să aibă obligatoriu regulile pentru **build** și **clean**. Regula **build** trebuie să aibă ca efect compilarea surselor și crearea binarului **stp**.

Rularea temei se va face astfel:

./stp initialise.in topology.in tasks.in stp.out , unde scopul fiecărui fișier a fost deja explicat la punctele anterioare.

Se pot folosi matrice pentru adiacență și pentru distanța dintre două noduri. Se poate folosi un vector pentru memorarea switch-urilor. Toate celelalte structuri în afară de acestea trei trebuie să fie realizate cu noțiuni studiate la SD (liste, stive, cozi, arbori, grafuri etc.). **NU** se acceptă alocare statică și nici variabile globale. Singura excepție de la această regulă este un buffer local de dimensiune maxim 100 pentru citirea din fișiere.

Atenţie!

Orice rezolvare care nu conţine structurile de date specificate nu este punctată.

Temele vor fi punctate doar pentru testele care sunt trecute pe vmchecker.

Nu lăsaţi warning-urile nerezolvate, deoarece veţi fi depunctaţi.

Dealocaţi toată memoria alocată pentru reţinerea informaţiilor, deoarece se vor depuncta pierderile de memorie.

Tema este individuală! Toate soluţiile trimise vor fi verificate, folosind o unealtă pentru detectarea plagiatului.