Poduri (bridges)

Mihai Budiu — mihaib+@cs.cmu.edu
http://www.cs.cmu.edu/~mihaib/

15 ianuarie 1999

Subiect: algoritmii folosiţi de bridge-urile transparente

Cunoştinţe necesare: cunoştinţe elementare despre funcţionarea calculatoarelor

Cuvinte cheie: bridge, adresă, reţea, interfaţă, arbore de acoperire, algoritm distribuit

Cuprins

Tehnologiile noi se insinuează încetul cu încetul în existenţa oamenilor, schimbînd pînă la urmă radical stilul lor de viaţă. În prima fază, o tehnologie revoluţionară nu face decît să înlocuiască tehnologii mai vechi, a căror funcţiune o poate îndeplini mai bine. Dar, cu timpul, îşi asumă tot felul de sarcini inexistente anterior, transformînd societatea în mod fundamental. Aşa s-au petrecut lucrurile, de pildă, cu curentul electric: la început acesta a preluat funcţiile iluminatului cu gaz, pentru că era mai ieftin şi mai sigur. Dar încetul cu încetul i-au fost găsite mii de aplicaţii;

societatea modernă de astăzi nu mai este de conceput fără existenţa sa: electricitatea a devenit o parte din infrastructură.

O revoluţie asemănătoare este în curs de desfăşurare: Internetul, reţeaua mondială, este noua tehnologie care va schimba faţa lumii; începuturile sale comerciale sunt cu numai cinci ani în urmă, şi evoluţia sa se află încă în prima fază, de "îmbunătăţire" a serviciilor existente. Maturizarea sa este rapidă, şi în viitor ne aşteaptă mutaţii neaşteptate.

Internetul este, aşa cum arată şi numele său, "inter-net", o *colecţie de reţele*, o super-reţea, care leagă laolaltă o sumedenie de reţele mici. Atenţia articolului de faţă se va îndrepta spre una din cărămizile constitutive ale Internet-ului, cea mai importantă (din punct de vedere cantitativ): reţeaua locală.

1 Reţele locale

În 1973 Robert Metcalfe, de la centrul de cercetare PARC (Palo Alto Research Center) al companiei Xerox, împreună cu alţi cercetători, punea la punct un tip de reţea extrem de ieftin şi eficace, numit Ethernet. Ethernetul permitea conectarea mai multor calculatoare printr-un singur cablu coaxial, asemănător cu cele folosite în televiziunea pe cablu; viteza de transmisiune iniţială era de pînă la 4 megabiţi pe secundă (adică în jur de jumătate de megaoctet pe secundă).

Pentru a comercializa Ethernetul, Xerox a format apoi împreună cu Intel şi Digital o nouă companie, condusă de Metcalfe, care a devenit apoi independentă sub numele "3 companii", sau 3Com. 3Com este la ora actuală, după Cisco, cea mai mare companie de echipamente de reţele din lume, cu venituri anuale de 6 miliarde de dolari.

Tipul de reţea inventat de Xerox a fost standardizat de IEEE sub sugestivul nume de 802.3; deşi reţeaua 802.3 este oarecum diferită de Ethernet, numele încetăţenit pentru aceste reţele este tot Ethernet. Ethernet-ul a fost apoi perfecţionat pentru a merge la viteze de pînă la 10Mbps. La începutul anilor '90 au apărut variantele fast-Ethernet, de 100Mbps, iar acum este în proces de standardizare varianta Gigabit-Ethernet, care merge la 1000Mbps! Ethernet-ul este de departe cea mai populară reţea din lume, ca număr de instalări. Ethernet-ul este ceea ce se numeşte o "reţea locală" (LAN: Local Area Network). Motivul principal este că poate conecta calculatoare pe o distanţă relativ restrînsă (lungimea unui cablu este limitată din motive de durată a propagării semnalului electromagnetic la 500m).

La începutul anilor '80, General Motors şi Boeing erau nesatisfăcute de anumite proprietăţi ale reţelei Ethernet (şi anume de faptul că Ethernet nu garantează cît de mult timp trece pînă cînd un calculator poate transmite un pachet), aşa că au investit pentru dezvoltarea unor tipuri alternative de reţele locale. Comitetul IEEE 802, care a standardizat Ethernet, s-a ocupat şi de aceste noi tipuri de reţele. Reţelele rezultate sunt 802.4, sau "token bus" şi 802.5, sau "token ring"[1]. Token bus foloseşte tot un cablu coaxial, însă ordinea accesului calculatoarelor (staţiilor) la cablu este riguros stabilită; în acest fel fiecare calculator ştie că după un anumit timp va avea posibilitatea să trimită ce are de trimis. Token ring conectează calculatoarele într-un inel fizic, fiecare calculator cu doi vecini; datele curg apoi de-a lungul inelului într-o singură direcţie. Cele trei tipuri fundamentale de reţele locale sunt ilustrate în figura 1.

[1] În ultima vreme au apărut şi standardele 802.6, pentru reţele de întindere medie (metropolitane), şi este în lucru standardul 802.11 despre reţele fără fir (wireless), cu transmisiune radio.

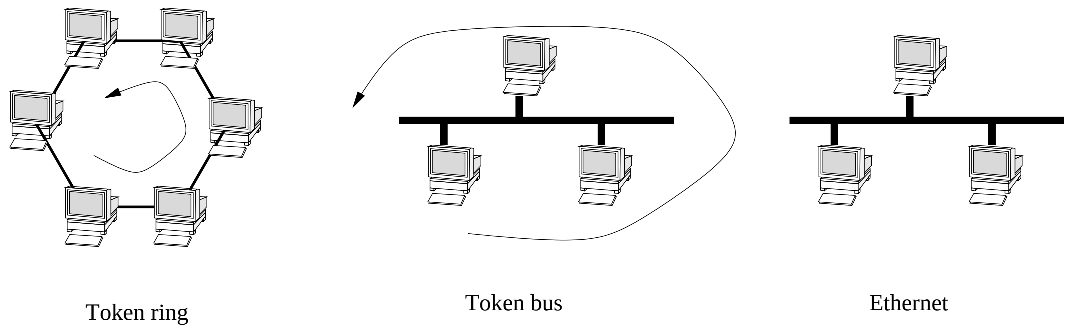

Figura 1: Cele trei tipuri de reţele locale. Token bus şi Ethernet au acelaşi tip de cablu, dar diferă prin ordinea de transmisiune, care este precis determinată pentru un token bus, dar este “primul care apucă” pentru un Ethernet.

În acest articol ne interesează mai puţin natura fizică a reţelelor şi mai mult conceptele care stau la baza utilizării lor. În unele privinţe toate reţelele standardului 802 se aseamănă, de aceea anumite părţi ale standardului sunt comune tuturor tipurilor; de pildă 802.1 stabileşte regulile de adresare pe toate aceste reţele.

1.1 Adrese la nivel de legătură

De fiecare dată cînd într-o conversaţie sunt implicate mai mult de două persoane, este necesar să se folosească nişte metode pentru a identifica fiecare participant. Fiecare participant trebuie să aibă un nume[2].

În terminologia reţelelor de calculatoare numele unui participant se numeşte *adresă*. În reţelele locale 802 adresele sunt numere de 48 de biţi (6 octeţi). Metoda standard de scriere a unei adrese foloseşte şase numere scrise în hexazecimal, separate de liniuţe. Aceasta este, de pildă, o adresă validă: 8-0-20-c0-ff-ee. IEEE are grijă ca în lume să nu existe două calculatoare cu aceeaşi adresă. Acest lucru se face atribuind fiecărui fabricant un anumit spaţiu de adrese (descris de primii 3 octeţi), fabricanţii apoi promit ca fiecare “placă” fabricată să aibă o altă adresă.

Unicitatea adreselor, şi faptul că sunt suficient de multe, face operarea şi administrarea reţelelor locale o treabă relativ simplă; în cele mai multe cazuri instalarea unui calculator într-o reţea locală este “plug-and-play”: nimic nu trebuie configurat de către un administrator uman pentru a-l face să comunice cu celelalte[3].

Cînd un calculator vrea să trimită date altui calculator, atunci pune pe reţea un pacheţel; în pacheţel se află, în afară de date, şi informaţii despre:

- Adresa destinaţie a datelor;
- Adresa sursă de unde au plecat datele;

[2] De fapt vom vedea mai tîrziu că un participant poate să aibă mai multe nume simultan, cîte unul pentru fiecare reţea la care este cuplat.

[3] Cuplarea unui calculator în Internet necesită oarecare configurare; configurarea însă se face pentru protocoalele de nivel superior TCP/IP: calculatorul funcţionează în reţeaua locală de îndată ce este conectat. Despre TCP/IP am scris un articol în PC Report din mai 1998; copia articolului este disponibilă în pagina mea de web.

- Informaţii despre cine este de aşteptat să consume datele la destinaţie (numărul protocolului). Asta pentru că pe un calculator se pot executa simultan mai multe programe de comunicaţii în reţea; numărul protocolului indică căruia anume îi sunt destinate datele.

Anumite adrese sunt rezervate pentru a indica grupuri de calculatoare (adrese *multicast*) sau "toate calculatoarele de pe reţeaua locală" (*broadcast*, difuzare). Un calculator poate fi instruit să asculte unele adrese de tip multicast; în acest fel o informaţie poate fi transmisă cu un singur pachet de la o sursă la mai multe destinaţii.

Transmisiunile de tip multicast sunt utile într-o multitudine de circumstanţe:

- Pentru transmisiuni de tip teleconferinţă în care participă mai multe calculatoare;
- Pentru a descoperi cine oferă anumite servicii. De pildă atunci cînd este pornit un calculator dintr-o reţea Novell, el va vrea să vadă cine anume este serverul de disc; atunci el va "zbiera" un pachet pentru toată lumea conţinînd întrebarea "care eşti server?". Pentru că acest pachet este trimis pe o adresă de broadcast, toate calculatoarele îl vor primi şi procesa, iar cel care este chiar serverul de disc va răspunde. Pentru a trimite răspunsul va folosi ca destinaţie chiar adresa sursă din pachetul "broadcast";
- Pentru a anunţa servicii interesante pentru toată lumea (ex.: server de disc);
- Sunt folosite de "poduri" (bridges) pentru a discuta între ele, după cum vom vedea mai jos.

2 Interconexiuni: repetoare şi poduri

Dacă avem două reţele locale pe care vrem să le conectăm între ele avem două posibilităţi. Cea mai simplă foloseşte un repetor (*repeater*).

Repetorul nu este nimic altceva decît un amplificator electric, care preia semnale dintr-o parte şi le pune în cealaltă. Reţelele de televiziune pe cablu folosesc astfel de amplificatoare, pentru a mări puterea semnalului transmis pe distanţe mari. După cum este evident, repetoarele se pot folosi numai între reţele locale de acelaşi tip (adică nu putem conecta un token bus cu un token ring).

Mai mult, deşi sunt eficace în a transforma două reţele într-una, în anumite privinţe reţeaua rezultată este limitată. De pildă, aşa cum am menţionat mai sus, durata de propagare a semnalului electromagnetic între oricare două calculatoare nu poate fi prea mare într-o reţea de tip Ethernet. O altă limitare provine din faptul că toate calculatoarele de pe aceeaşi reţea locală folosesc în comun acelaşi mediu de transmisie, deci cu cît sunt mai multe, cu atît rata efectivă de transmisie de care se poate bucura fiecare din ele scade mai mult.

Pentru a rezolva acest gen de probleme au fost create podurile (bridges).

2.1 Poduri multi-protocol

Un pod nu este nimic altceva decît un calculator specializat care este conectat la două (sau mai multe) reţele locale simultan. Podul ştie să preia pachete de date dintr-o reţea şi să le transmită în cealaltă, dacă este nevoie. Poduri sunt ilustrate în figurile 2 şi 3.

Aparent funcţiunile unui pod sunt aceleaşi cu ale unui repetor. Există însă importante diferenţe:

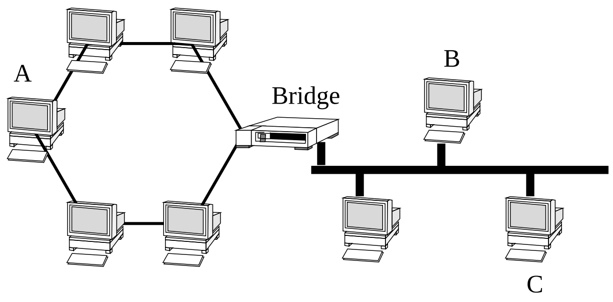

Figura 2: Un pod conectat între două reţele: un token-ring şi un Ethernet.

- Atunci cînd un pod leagă două reţele locale de tipuri diferite, el face şi conversii între cele două reţele; pachetele arată altfel, şi regulile de transmisiune sunt diferite. Un astfel de pod se numeşte pod multi-protocol, pentru că pe fiecare parte vorbeşte un protocol diferit;
- Un pod nu copiază electric informaţii de pe o parte pe alta; el *citeşte, stochează şi înaintează* pachetele de date (*store and forward*). Asta are nişte consecinţe foarte importante: de exemplu dacă pe reţeaua din stînga se află un pachet cînd un calculator din dreapta transmite, atunci podul poate captura pachetul din dreapta şi îl poate stoca pînă reţeaua din stînga devine liberă;
- Podul nu copiază orbeşte pachete dintr-o parte într-alta; dacă un pod determină că un pachet are atît sursa cît şi destinaţia de aceeaşi parte, atunci nu copiază pachetul de loc în cealaltă parte. Astfel, pe ambele reţele legate de un pod se pot afla simultan pachete diferite; acest lucru nu este adevărat despre un repetor. Prin această proprietate podul măreşte capacitatea de transport a reţelelor pe care le conectează (comparat cu o singură reţea cu acelaşi număr de calculatoare).

3 Podurile transparente

Lucrurile par să stea ca şi cum transmisiunea unui pachet între calculatoarele A şi B din figura 2 se face în doi paşi: de la A la bridge, şi de la bridge la B. Pentru a lega însă cele două reţele într-un mod invizibil, ca şi cum ar fi una singură, existenţa podului trebuie să fie indiscernabilă; podul trebuie să fie *transparent*.

Din cauza asta, podul capturează pachetul trimis de A, deşi acesta specifică adresa B (care nu este tot una cu adresa podului). Podul apoi trimite pachetul pe reţeaua Ethernet fără a schimba însă adresa sursă sau destinaţie din pachet! Pachetul apare pe reţeaua Ethernet ca şi cum A ar fi fost cuplat direct la reţea.

Întrebarea care se pune este: din moment ce podurile nu necesită nici un fel de configuraţie manuală, de unde ştiu ele de fapt cînd trebuie să preia un pachet şi cînd nu, şi de unde ştiu pe care dintre reţele să-i dea drumul?

3.1 Algoritmul de învăţare

Răspunsul este: nici nu ştiu, cel puţin la început. Însă învaţă; din cauza asta se şi numesc *learning bridges*. Vom ilustra algoritmul printr-un exemplu:

1. Să presupunem că A vrea să trimită un pachet calculatorului B. Atunci el crează un pachet cu adresa sursă A, adresa destinaţie B şi îl pune pe reţeaua lui locală, token ring.
2. Pachetul eventual ajunge la bridge. Bridge-ul observă adresa sursă şi zice: "aha, iată un pachet de la A care vine din reţeaua token-ring; asta înseamnă că A este de fapt aflat în reţeaua token-ring!"
3. Bridge-ul zice apoi: "destinaţia este B. N-am nici cea mai mică idee unde e B. Hai să copiez pachetul în toate reţelele la care sunt eu conectat; în felul ăsta dacă există o cale pînă la B, pachetul va ajunge." Zis şi făcut; avem acum două copii ale pachetului de la A, una în reţeaua token-ring şi una în Ethernet.
4. B primeşte pachetul. După o vreme probabil B răspunde cu un pachet pentru A. De data asta podul ştie că A este în reţeaua cealaltă, aşa că face o nouă copie a pachetului. În plus podul află adevărata adresă a lui B: în reţeaua Ethernet.
5. Dacă mai tîrziu C trimite pentru B un pachet, podul nu mai face nici o copie, pentru că ştie deja că C şi B sunt în aceeaşi reţea. În plus podul află despre poziţia lui C.

Podurile "uită" periodic informaţiile învăţate; în acest fel o staţie poate fi deconectată dintr-o reţea şi mutată în cealaltă, şi podurile continuă să opereze corect.

4 Buclele în topologie

Este uşor de verificat că algoritmul descris mai sus funcţionează şi pentru poduri care leagă mai mult de două reţele, şi că funcţionează şi pentru reţele cu topologii complexe, ca în figura 3.

Din păcate lucrurile nu stau chiar aşa pentru orice topologie; figura 4 prezintă o topologie de reţele locale pentru care aplicarea acestui algoritm duce la o catastrofă.

Să vedem tot printr-un exemplu ce se întîmplă de fapt:

1. A trimite un pachet pentru B;
2. Ambele bridge-uri văd pachetul de la A şi marchează poziţia curentă a lui A în reţeaua de sus;
3. Să zicem că bridge-ul 1 copiază primul pachetul în reţeaua de jos;
4. Bridge-ul 2 vede acest pachet de la A (pentru că adresa sursă nu este schimbată de un pod) şi presupune că A s-a mutat în reţeaua de jos. Aşa că marchează noua adresă a lui A, după care copiază "noul" pachet în reţeaua de sus (un bridge nu memorează pachetele pe care le-a prelucrat; pentru acest bridge ceea ce a apărut este un nou pachet transmis de A);

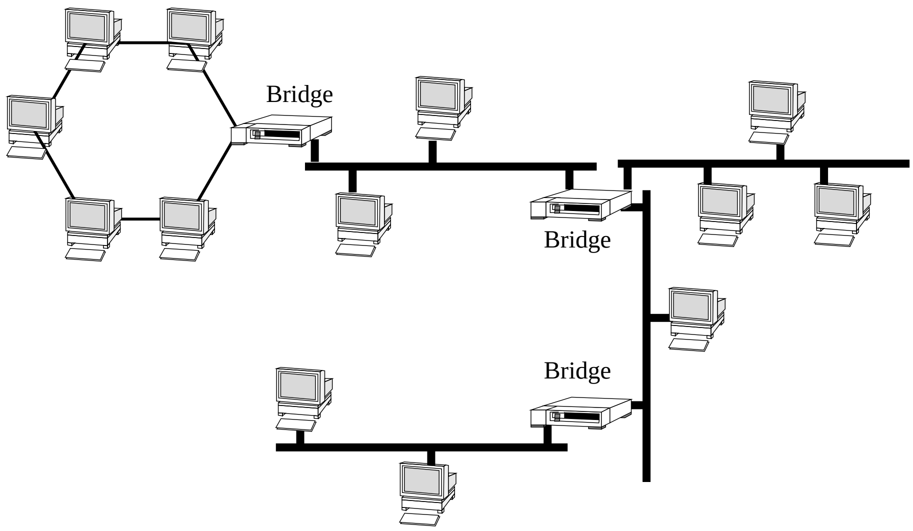

Figura 3: O topologie complexă pentru care algoritmul de învăţare al podurilor transparente se comportă totuşi corect.

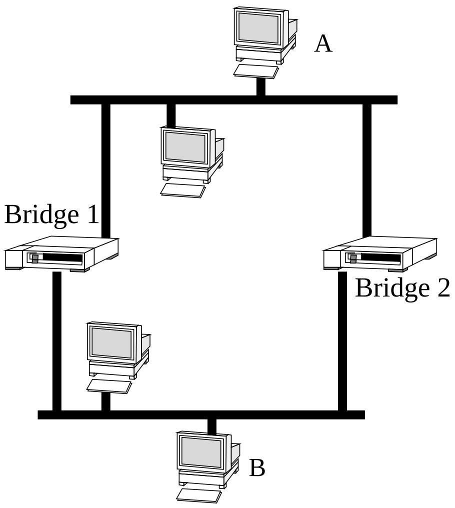

Figura 4: O topologie pentru care algoritmul de învăţare aşa cum este prezentat în text eşuează datorită prezenţei ciclurilor.

5. Şi tot aşa, povestea se repetă. Nu numai că pachetul de la A este transmis într-una între cele două reţele, el se şi multiplică. Cum transmisiunea unui pachet e o chestiune de cîteva milisecunde, într-o secundă reţeaua este inundată cu copii ale aceluiaşi pachet şi devine complet nefuncţională (nu mai poate transporta nici un fel de trafic util).

În general, orice reţea care are cicluri (mai multe drumuri posibile între două calculatoare) duce la astfel de probleme, pentru că bridge-urile nu se aşteaptă să vadă pachete de la un calculator venind de pe două interfeţe diferite.

5 Arborele de acoperire

Care e soluţia? Putem cere administratorilor de reţea să nu conecteze niciodată reţele cu cicluri; în acest fel sunt singurii vinovaţi dacă aşa ceva se întîmplă.

Însă ideea de a conecta o reţea în mod redundant, în aşa fel încît între calculatoare să existe uneori mai mult de un singur drum, este o idee foarte bună, pentru că este *rezilientă la erori*. Dacă avem două conexiuni şi una dintre ele se strică, cealaltă poate prelua traficul fără întreruperea conectivităţii.

Clar, trebuie oferită o altă soluţie. Aceasta constă în algoritmul prin care bridge-urile reduc orice reţea cu cicluri la una fără cicluri, căzînd de acord ca anumite legături să nu fie niciodată folosite. În cazul în care topologia reţelei se schimbă, prin căderea uneia dintre legături, bridge-urile recalculează legăturile folosite, refăcînd conectivitatea reţelei.

Dacă vedem o serie de reţele LAN ca noduri într-un graf, iar bridge-urile ca arce, algoritmul care rezolvă problema expusă mai sus calculează ceea ce se numeşte *arbore de acoperire* (*spanning tree*) al grafului. Acesta este un graf care este arbore (nu are cicluri) şi cuprinde toate nodurile grafului. Acest algoritm a fost inventat de Radia Perlman, şi este folosit de toate podurile transparente din lume. Prezentarea algoritmului se bazează pe cea din cartea ei de la Addison-Wesley "Interconnections", pe care am recomandat-o şi cu alte ocazii. Radia Perlman este o personalitate în lumea reţelelor; are un doctorat în domeniu de la MIT, şi a lucrat mai bine de 10 la Digital în proiectare de protocoale. Multe din protocoalele de rutare din Internetul de azi sunt bazate pe prototipurile folosite în DECnet (reţeaua dezvoltată de Digital); de pildă protocolul OSPF (Open Shortest-Path First) este în mare măsură creaţia Radiei Perlman.

Acest gen de algoritmi sunt foarte greu de înţeles, şi mai greu de proiectat, extrem de greu de depanat şi aproape imposibil de certificat ca fiind corecţi. Asta pentru că aceşti algoritmi sunt *distribuiţi*: fiecare entitate care participă în calcule are numai informaţii parţiale despre întregul ansamblu, şi trebuie să reconstituie totul doar din informaţiile primite. Vom vedea mai jos că o sumedenie de factori practici (cum ar fi căderile unor staţii, erori tranziente, etc.) fac garantarea funcţionării unui astfel de protocol în *toate* circumstanţele o treabă practic imposibilă. Dar înainte de asta să vedem care este algoritmul de bază.

5.1 Algoritmul de votare

Algoritmul se bazează pe un set de mesaje de configurare pe care podurile le trimit între ele, folosind o adresă multicast specială. Folosind mesajele de configurare, toate podurile ajung la o aceeaşi concluzie despre topologia reţelei:

- Toate aleg un "pod rădăcină", de la care începe calculul arborelui de acoperire;
- Fiecare pod calculează cea mai scurtă "distanţă" (într-un sens care va fi precizat mai tîrziu, care depinde şi de numele podurilor prin care trec mesajele, nu numai de distanţa fizică) de la sine la podul rădăcină;
- Pentru fiecare reţea locală se alege un pod reprezentativ (*designated bridge*). Acesta este podul de pe LAN care este cel mai apropiat de rădăcină. Toate pachetele de pe acest LAN spre rădăcină vor fi transmise numai de podul ales;

- Fiecare pod alege un port (o interfaţă, o conexiune la o reţea) care este cel mai apropiat de podul rădăcină;
- Toate porturile selectate formează laolaltă arborele de acoperire.

Fiecare bridge transmite mesaje de configurare şi calculează în funcţie de mesajele primite cum arată reţeaua. Un pod nu retransmite mesajele primite de la alte poduri (aşa cum face cu mesajele care conţin date); fiecare mesaj de configurare deci călătoreşte pe un singur LAN. După un număr de runde, fiecare pod are suficiente informaţii pentru a deduce topologia corectă şi algoritmul se termină.

Fiecare bridge foloseşte drept nume (identificator, ID) un număr unic, de obicei adresa uneia din interfeţele sale (întotdeauna aceeaşi). Pod-rădăcină va fi ales cel care are cel mai mic ID.

Fiecare pod transmite un mesaj compus din trei bucăţi:

1. Cel mai mic ID de care a auzit vreodată de la vreun vecin;
2. Costul celei mai scurte cărări de la sine la podul cu cel mai mic ID;
3. Propriul ID.

Iniţial fiecare pod crede despre sine că este rădăcina, deci primul mesaj are forma (ID meu, 0, ID meu). Cu vremea această opinie se schimbă pentru fiecare pod. Un bridge primeşte într-una mesaje de configurare pe fiecare port, şi menţine *pentru fiecare port* "cel mai bun" mesaj de configurare pe care l-a primit. "Cel mai bun" mesaj de configurare este obţinut comparînd:

1. Mesajele primite pe acel port;
2. Mesajele pe care podul se pregăteşte să le transmită pe acel port.

Un mesaj e "mai bun" decît altul dacă e "mai mic" în ordine lexicografică. Cu alte cuvinte: $(a, b, c) < (d, e, f)$ dacă:

1. $a < d$ sau
2. $a = d$ şi $b < e$ sau
3. $a = d$ şi $b = e$ şi $c < f$.

Podurile transmit simultan mesaje de configurare pe toate porturile. În momentul în care un pod primeşte pe un port un mesaj mai bun decît ar fi vrut el să transmită, atunci încetează să mai transmită pe acel port. Motivul este că mesajele pot deveni doar din ce în ce mai bune cu trecerea timpului, deci propriul lui mesaj nu mai are nici o şansă să devină cel mai bun pe acel LAN.

Un bridge îşi clasează apoi porturile în trei categorii:

- Portul dintre rădăcină; pe acest port bridge-ul a primit cel mai bun mesaj. Acest port va fi activ (folosit) pentru toate comunicaţiile care merg/vin dinspre rădăcină;
- Porturile pe care bridge-ul a trimis el însuşi cel mai bun mesaj. Pentru toate LAN-urile legate la aceste porturi acest bridge este "designated bridge". Aceste porturi sunt şi ele active.

- Toate celelalte porturi sunt blocate (inactive); pe ele nu vor fi transmise niciodată mesaje, atîta vreme cît topologia reţelei nu se schimbă.

În exemplul din figura 4, rezultatul protocolului va fi că bridge-ul cu cel mai mic ID devine rădăcină, iar celălalt îşi dezactivează ambele porturi.

Figura 5 ilustrează un fragment dintr-o reţea locală şi cele trei feluri de porturi pentru un bridge b2.

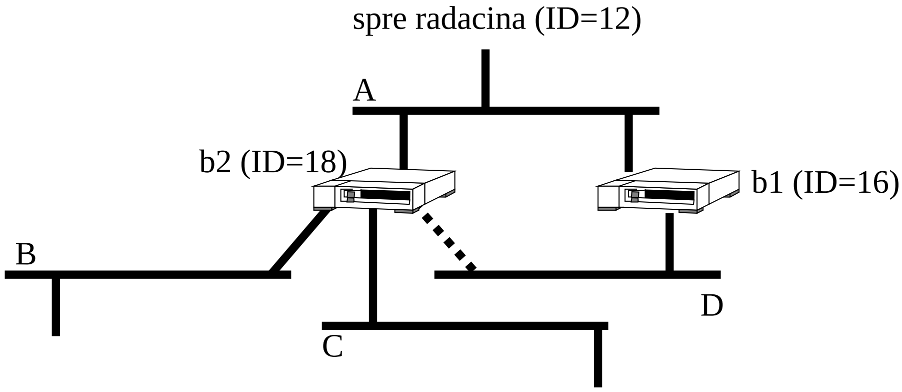

Figura 5: Fragment dintr-o conexiune de reţele locale. b1 este pod desemnat pentru LAN-ul D, iar b2 este pod desemnat pentru LAN-urile B şi C. Legătura (portul) lui b2 la D este inactivă. Portul lui b2 la A este portul lui b2 spre rădăcină.

5.2 Un exemplu numeric

Iată concret pe un exemplu numeric acţiunile lui b2 din figura 5 într-o rundă posibilă a protocolului:

Mesajele primite de b2 (ID = 18)

	Rădăcină	Cost	Transmiţător
Port spre A	12	4	85
Port spre B	20	0	20
Port spre C	22	2	44
Port spre D	12	5	16

Cel "mai bun" mesaj primit este (12, 4, 85). Ca atare b2 decide că el se află la distanţă 4+1=5 de rădăcină; mesajul pe care îl va transmite în runda următoare va fi deci: (12, 5, 18). Acest mesaj este comparat cu toate celelalte 4 mesaje. Acest mesaje este mai bun decît cele de la porturile spre B şi C, deci b2 devine pod desemnat pentru aceste LAN-uri. Portul spre A duce spre rădăcină (pe acolo a sosit cel mai bun mesaj), iar portul D este dezactivat, pentru că mesajul primit de acolo e mai bun decît cel calculat. Rezultatul algoritmului în jurul lui b2 este cel din figura 5.

6 Rafinamente

Acesta este algoritmul distribuit pentru calculul arborilor de acoperire. Analizîndu-l cu oarecare grijă putem să ne convingem de corectitudinea lui: algoritmul se termină pentru că mesajele trimise vor fi din ce în ce mai "bune", pînă cînd numai bridge-ul rădăcină mai vorbeşte, topologia stabilită nu va avea cicluri, pentru că cel puţin un bridge într-un ciclu va fi forţat să-şi dezactiveze o interfaţă, şi topologia stabilită va fi conexă (adică va lega toate reţelele), pentru că fiecare reţea o să aibă un "designated bridge", şi pentru că orice lipsă de conectivitate între două părţi este remediată de o rundă de mesaje. (Desigur, aceasta este o argumentaţie foarte sumară.)

Pe de altă parte această demonstraţie este valabilă în cazul în care reţeaua este stabilă (*stable, quiescent*); în momentul cînd se fac schimbări topologice ce se întîmplă cu algoritmul? Dacă un bridge sau o reţea cad, sau dacă administratorul adaugă noi bridge-uri, care încă nu ştiu de vechea configuraţie, cum de lucrurile continuă să funcţioneze?

Aici lucrurile sunt mult mai problematice, tocmai pentru că e foarte greu de imaginat dinainte orice tip de eveniment; pentru unele cazuri foarte exotice, cum ar fi prezenţa unei defecţiuni care permite transmisiunea într-un singur sens, nici nu există soluţii prestabilite.

Iată ilustrate unele dintre posibilele complicaţii şi soluţiile adoptate în practică:

Căderi: fiecare bridge "învecheşte" (*ages*) mesajele primite; cînd acestea ating senectutea, bridge-ul le uită complet şi re-iniţiază un calcul al topologiei. În timpul unei funcţionări normale, bridge-ul rădăcină trimite periodic mesaje de configurare (numite "hello"), care cauzează fiecare pod să reconfigureze. Dacă anumite mesaje nu mai apar din cauza unei căderi, vechea configuraţie se învecheşte şi atunci cînd la cel puţin un pod dispare complet, acel pod iniţiază o nouă execuţie completă a algoritmului, pentru a descoperi noua topologie. Chiar alegerea acestor doi parametri: distanţa dintre hello şi timpul de învechire, este o problemă foarte spinoasă: valori mici supra-încarcă reţeaua cu mesaje inutile în timpul funcţionării normale, dar valori prea mari fac descoperirea erorilor prea înceată. Valorile recomandate sunt de 2, respectiv 20 de secunde.

Evitarea buclelor temporare: apariţia unor bucle este o catastrofă, şi trebuie evitată cu orice preţ. Pentru ca un pod nou adăugat să nu creeze o buclă, funcţionarea unui pod este împărţită de standardul 802.1 în două etape:

- O etapă de învăţare, în care bridge-ul nu retransmite nici un fel de pachete de date, pentru a nu cauza cicluri. Singura funcţiune este de a schimba mesaje de configurare pentru a învăţa topologia;
- Etapa de funcţionare propriu-zisă, în care datele sunt transmise conform cu arborele ales.

Conectivitate unidirecţională: E posibil ca hardware-ul unui port să se strice, şi ca un bridge să fie incapabil să primească pachete de la un anumit LAN, deşi poate transmite perfect. În acest caz bridge-ul se va declara "designated bridge" pentru acel LAN, ceea ce ar putea fi o decizie incorectă. Standardul nu specifică ce trebuie făcut în acest caz; singura soluţie este detectarea şi repararea erorii de către un operator uman.

7 Concluzii

Închei aici acest articol despre poduri, deşi problemele care trebuie considerate în practică sunt mult mai multe şi mai complicate decît am apucat să ilustrez. Principiile de bază ale funcţionării unui bridge sunt însă, sper, cuprinse în acest text. Restul sunt detalii.

Să recapitulăm învăţămintele principale: există mai multe tipuri de reţele locale, care realizează transmisiunea informaţiei între calculatoare relativ puţine (sute) aflate la distanţe relativ mici (kilometri) cu viteza relativ mare (sute de megabiţi pe secundă, în tehnologiile cele mai mature). Reţelele locale se pot lega între ele folosind repetoare sau poduri. Repetoarele sunt doar o conexiune electrică, pe cînd bridge-urile sunt echipamente inteligente, care învaţă aşezarea staţiilor, filtrează traficul nenecesar între reţelele pe care le leagă, şi colaborează pentru a construi o reţea completă fără cicluri, pentru a preveni proliferarea nelimitată a unor pachete.

Reţelele locale reprezintă cea mai importantă (numeric vorbind) piesă din Internet, care este o colecţie de reţele legate laolaltă. Evoluţia lor este departe de a fi terminată: cursa spre distanţe şi viteze mari continuă. Fiţi cu ochii, şi cu datele, pe ele!