

UNIVERSITATEA "POLITEHNICA" DIN BUCUREȘTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI
TEHNOLOGIA INFORMAȚIEI

Matei Dan CIOBOTARU

**Caracterizarea, gestionarea și monitorizarea rețelelor de
calculatoare ale sistemului de achiziție de date ATLAS**

**Characterizing, managing and monitoring the networks
for the ATLAS Data Acquisition System**

Rezumatul tezei de doctorat

Conducător științific:

Prof. Vasile Buzuloiu

2007

Cuprins

1	Introducere	3
1.1	Structura tezei	4
2	Sistemul pentru achiziția datelor	4
2.1	Arhitectura rețelei TDAQ	5
3	Testarea echipamentelor Ethernet	7
3.1	Platforma GETB	7
3.2	Sistemul de testare	12
4	Analiza traficului din rețeaua TDAQ	14
4.1	Viteza la recepție	14
4.2	Gradul de ocupare al cozilor	16
4.3	Aplicații	17
5	Unelte pentru gestiunea rețelelor	17
5.1	Configurarea dispozitivelor	17
5.2	Descoperirea topologiei rețelei	18
6	Monitorizarea traficului prin eșantionare	19
6.1	Valori estimative	20
6.2	Standardul sFlow	20
6.3	Exemplu de aplicație	20
7	Încheiere	22
7.1	Contribuții	22
7.2	Perspective	23
	Bibliografie selectivă	23

1 Introducere

Fizica atomică este o ramură a fizicii ce studiază particulele din care este alcătuită materia. O parte dintre acestea pot fi observate numai în urma coliziunilor ce au loc în interiorul acceleratoarelor de particule. În 2008, la CERN¹ va fi finalizat un nou accelerator circular numit *the Large Hadron Collider (LHC)*, având o circumferința de 27 Km. LHC-ul va fi gazda a șase detectoare de particule (Figura 1.1).

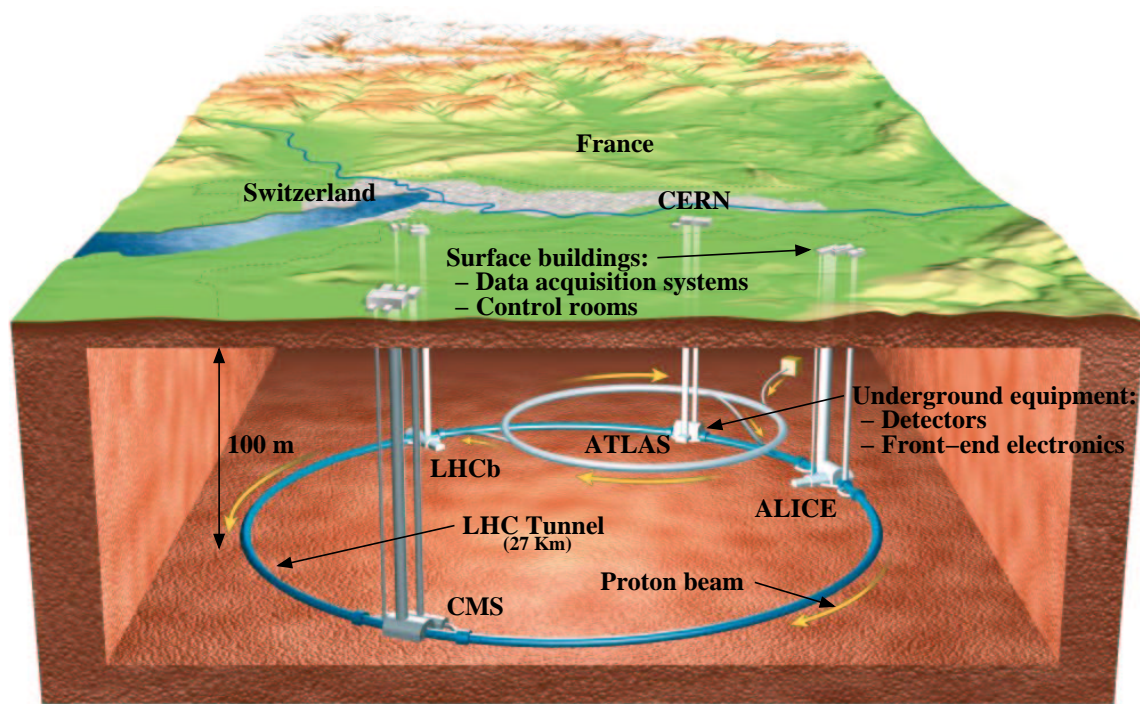


Figura 1.1: Acceleratorul LHC de la CERN.

ATLAS (*A Toroidal LHC ApparatuS*) [1] este unul dintre acestea. Rolul său este de a observa ciocnirile dintre protoni (numite *evenimente*) și de a furniza informații despre cele care dau naștere unor fenomene fizice deosebite: particule rare, abateri de la modelele teoretice existente, etc. Bozonul Higgs este un exemplu de particulă a cărei existență încă nu a fost dovedită experimental. Descoperirea acesteia ar ajuta la înțelegerea conceptului fizic de masă.

Sistemul ATLAS trebuie să prelucreze evenimente ce au loc la o rată de 40 MHz, aceasta fiind rata la care se produc coliziunile în LHC. Cantitatea de date asociată unei coliziuni este de 1.6 Mbyte. Prin urmare, ATLAS-ul generează un volum de 64 Tbyte de date în fiecare secundă.

Deoarece tehnologiile actuale nu permit stocarea în timp real la o asemenea viteză, detectorul ATLAS este conectat la un sistem de filtrare și de achiziție de date: *Trigger and Data Acquisition, TDAQ*.

Acest sistem analizează în timp real fiecare eveniment și le elimină pe cele care nu conțin informații utile². Rata la ieșirea sistemului TDAQ este de numai 300 Mbyte/s. Evenimentele care au trecut de filtrele din TDAQ sunt arhivate și studiate ulterior în amănunțime. Procesarea datelor în cadrul TDAQ se face pe câteva mii de calculatoare interconectate într-o rețea locală de mare viteză.

¹CERN este un acronim pentru "Conseil Européen pour la Recherche Nucléaire". Institutul este situat la granița dintre Elveția și Franța, lângă Geneva.

²Evenimentele "interesante" sunt relativ rare – în medie, apare unul la un milion.

1.1 Structura tezei

În această lucrare vorbim despre implementarea rețelei de calculatoare care stă la baza sistemului TDAQ. Teza este alcătuită din trei părți – acestea corespund celor trei faze care au fost necesare finalizării rețelei TDAQ. Autorul a contribuit la fiecare dintre aceste etape de dezvoltare, după cum vom vedea în cele ce urmează:

Partea I – Proiectarea Sistemul de achiziție TDAQ necesită o rețea de calculatoare de foarte mare viteză (vezi Capitolul 2). Cerințele acesteia cu privire la performanță sunt cu mult mai ridicate decât cele ale rețelelor obișnuite. Faza de proiectare a început cu un studiu al celor mai rapide tehnologii existente în domeniul rețelelor locale. În 2001 s-a ajuns la concluzia că Gigabit Ethernet este cea mai potrivită tehnologie pentru TDAQ. La puțin timp după aceea, a fost propusă o arhitectură completă a rețelei. Perioada care a urmat a fost dedicată alegerii celor mai potrivite dispozitive care să fie folosite la construcția acesteia. A fost necesar un studiu de piață în decursul căruia au fost evaluate numeroase echipamente de rețea. Un sistem complet de testare a fost dezvoltat în acest scop (una din principalele contribuții ale autorului, descrisă în Capitolul 3). De asemenea, tot ca parte a fazei inițiale de proiectare, s-a efectuat un studiu al caracteristicilor traficului prezent în rețeaua ATLAS. În Capitolul 4 prezentăm un model analitic al acestui tip de trafic; modelul a fost validat în mod experimental.

Partea a II-a – Instalarea Finalizarea fazei de proiectare a coincis cu începerea achiziționării echipamentelor, urmată de instalarea acestora. Fiecare dispozitiv trebuia să fie configurat; fiecare conexiune la rețea, verificată și documentată. Dimensiunile rețelei TDAQ au făcut necesară automatizarea acestor sarcini. În Capitolul 5 discutăm problema gestiunii unei rețele, după care descriem soluțiile propuse de autor în acest domeniu: un sistem care facilitează configurarea echipamentelor și un mecanism pentru descoperirea topologiei într-o rețea (a legăturilor fizice dintre dispozitivele de conectare).

Partea a III-a – Darea în funcțiune Se preconizează că detectorul ATLAS va începe să funcționeze în 2008. Din acel moment, se presupune că rețeaua va funcționa în mod ireproșabil. Toate echipamentele și toate legăturile vor fi supravegheate în permanență pentru a detecta și soluționa rapid eventualele defecțiuni. În Capitolul 6 prezentăm o tehnologie care ajută la diagnosticarea problemelor cauzate de traficul din rețea. Vom descrie standardul oficial pe care se bazează, precum și un exemplu de aplicație practică.

În cele ce urmează vom face o trecere prin toate aceste faze, concentrându-ne atenția asupra contribuțiilor personale. Ultimul capitol (al 7-lea) prezintă pe scurt cele mai importante realizări ale autorului. Activitatea pe care se bazează această lucrare s-a desfășurat la CERN în cadrul grupului de cercetări în domeniul rețelelor de calculatoare.

2 Sistemul pentru achiziția datelor

În ATLAS, sistemul de achiziție (TDAQ) trebuie să reducă rata de evenimente de la 40 MHz (64 Tbyte/s) la 200 Hz (300 Mbyte/s). Viteza de la ieșirea sistemului este impusă de limitele tehnologiilor de stocare. Pentru a obține o reducere a ratei de 1:200000, s-a optat pentru o filtrare în trepte, folosindu-se un lanț de prelucrare organizat pe trei niveluri. Pe fiecare nivel se face un compromis între timpul alocat analizei și acuratețea acesteia.

Primul nivel de filtrare (*Level 1 Trigger*) Acest nivel este bazat pe componente electronice realizate în mod special pentru ATLAS. Acestea sunt conectate direct la detector și procesează în timp real toate evenimentele la rata maximă de 40 MHz. Nivelul 1 lucrează cu algoritmi simpli și foarte rapizi. Rata de la intrare trebuie redusă cu un factor de 1:400, până la 100 kHz sau 160 Gbyte/s.

Memoria tampon Datele selecționate de Nivelul 1 sunt trimise către un set de memorii tampon numite *Read Out Buffers (ROB)*. ROB-urile, implementate pe plăci PCI, sunt instalate în calculatoare denumite *Read Out Systems (ROSs)*. Fiecare ROB este legat direct la un sub-detector din ATLAS. Un ROB conține doar o fracțiune din datele referitoare unui anumit eveniment. Informația completă asociată evenimentului este distribuită în ≈ 1600 ROB-uri (dispuse în ≈ 150 de ROS-uri).

Al doilea nivel de filtrare (*Level 2*) Acesta rulează în software pe un set de ≈ 500 de calculatoare numite *Level 2 Processing Units (L2PUs)*. Analiza de Nivel 2 folosește în medie doar 1% din informația din ROS-uri. Un L2PU are la dispoziție 10 ms pentru a extrage date din câteva ROB-uri și pentru a se decide dacă un eveniment trebuie acceptat. Rata maximă la ieșirea din Nivelul 2 este de 3.3 kHz.

Sistemul *Event Builder* Responsabilitatea acestuia este să adune toate datele referitoare la evenimentele care au fost validate de Nivelul 2. O aplicație software numită *Sub-Farm Interface (SFI)* colectează cele 1600 de fragmente din toate ROB-urile. Evenimentele complete sunt păstrate temporar în SFI-uri pentru a fi livrate mai departe ultimului nivel de filtre.

Al treilea nivel de filtrare (*Event Filter, EF*) Nivelului 3 îi sunt alocate 1600 de calculatoare dual-procesor. O stație EF (*Event Filter Processor, EFP*) preia câte un eveniment de la un SFI și îl prelucrează în aproximativ o secundă. Evenimentele “interesante” din punct de vedere al fizicii sunt trimise cu viteza de 320 Mbyte/s către unitățile de stocare pe bandă magnetică.

Rețeaua TDAQ asigură legăturile între elementele de procesare (L2PU-uri, EFP-uri) și nodurile care păstrează datele (ROS-uri și SFI-uri). Traficul este de tip “cerere-răspuns”. Rețeaua trebuie să transfere datele la o rată de aproximativ 150 Gbit/s, cu întârzieri minime și, de preferință, fără pierderi semnificative¹.

2.1 Arhitectura rețelei TDAQ

Studii efectuate în perioada 2001-2003 au demonstrat că tehnologia Ethernet reprezintă cea mai bună alegere pentru rețeaua TDAQ [2], [3]. Din punct de vedere al costurilor și al performanței, s-a dovedit a fi superioară altor tehnologii (ATM, de exemplu). Au urmat o serie de propuneri referitoare la arhitectura rețelei [4], o variantă definitivă fiind publicată în 2006 [5].

Rețeaua va fi organizată sub formă de arbore (Figura 2.1). Calculatoarele (care se instalează în rack-uri) sunt conectate la switch-uri concentratoare. Acestea sunt legate mai departe la switch-urile centrale. La nivelul unui rack se folosesc cabluri de cupru (UTP) și viteze de 1 Gbit/s. Conexiunile cu switch-urile centrale sunt bazate pe cabluri de fibră optică și lucrează la 10 Gbit/s.

Rețeaua TDAQ este împărțită din punct de vedere logic în trei subrețele (Figura 2.2). Două dintre ele sunt dedicate transferurilor de date: subrețeaua *FrontEnd* pentru traficul generat de nivelul 2 de filtrare și subrețeaua *BackEnd* pentru nivelul 3. A treia rețea este folosită pentru controlul aplicațiilor și pentru monitorizarea întregului sistem.

¹Sunt tolerate pierderi de pachete de ordinul 10^{-6} .

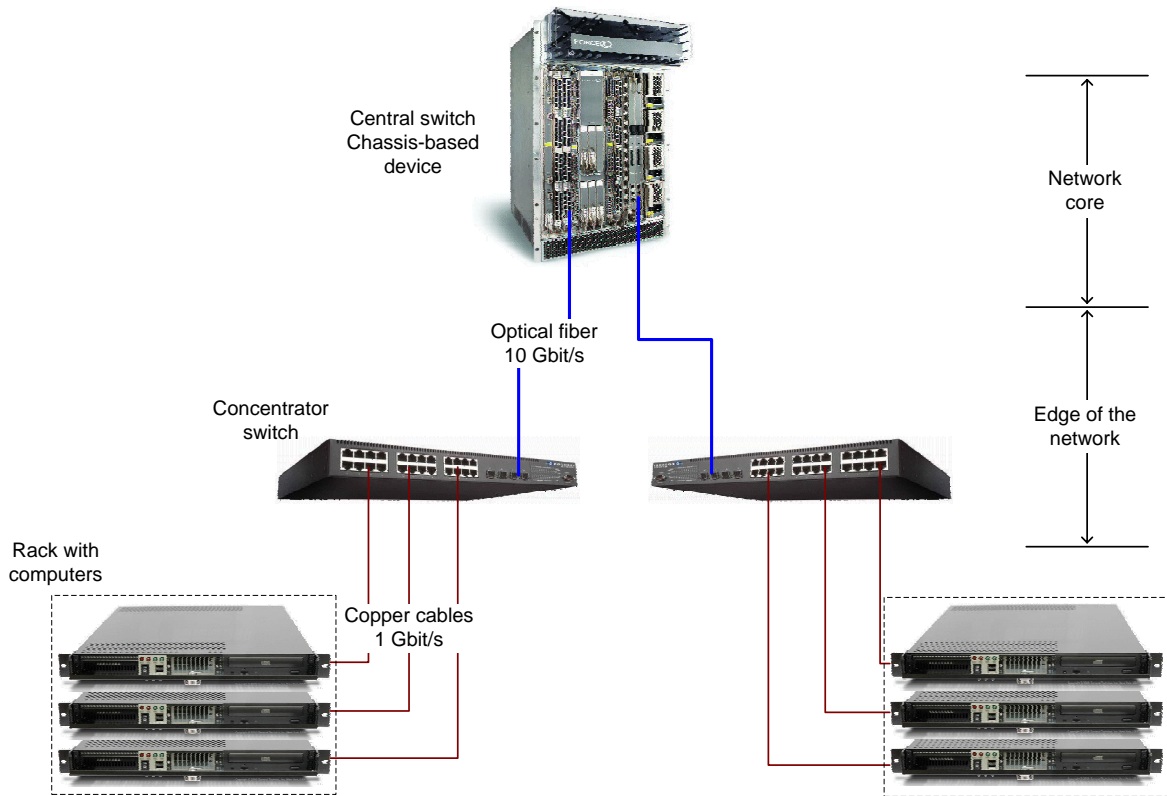


Figura 2.1: Modul de organizare al echipamentelor în rețeaua TDAQ.

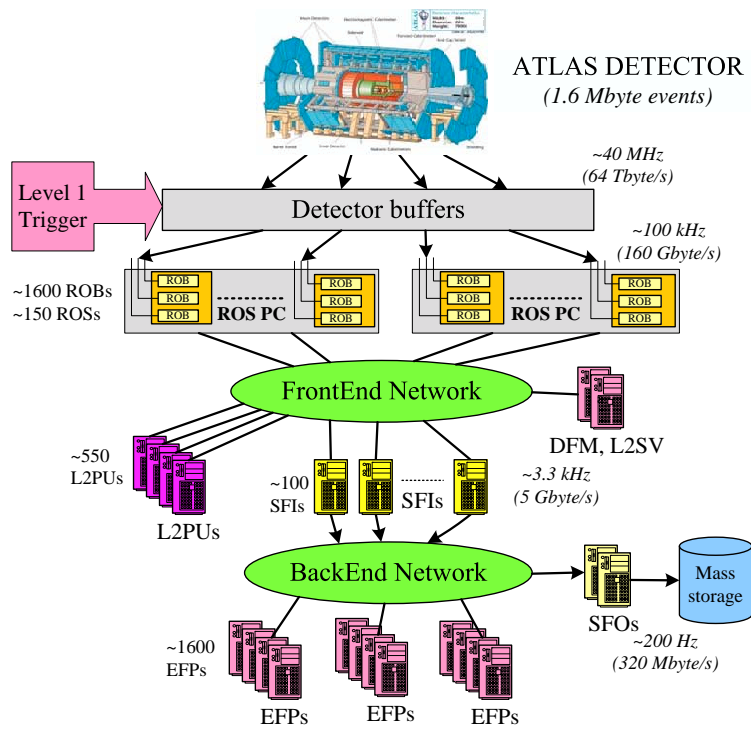


Figura 2.2: Rețele pentru transportul datelor în TDAQ.

Vor exista cel puțin două switch-uri centrale. Acestea vor fi conectate și configurate astfel încât să opereze în regim de redundanță. Defectarea unuia dintre ele nu va întrerupe întreg sistemul TDAQ (îi va reduce totuși viteza la jumătate).

2.1.1 Cerințe

Sistemul de achiziție TDAQ are caracteristicile unui sistem în timp cvasi-real cu limite de latență foarte stricte (cauzate de folosirea unui protocol de comunicație de tip cerere-răspuns). În cazul unor pierderi în rețea, aplicațiile de filtrare trebuie să recupereze fragmentele pierdute emițând noi cereri: performanța scade în mod simțitor în astfel de cazuri. În plus, protocolul de transport utilizat în ATLAS va fi UDP (din motive de eficiență). Acesta, spre deosebire de TCP, lasă gestionarea congestiei din rețea în seama aplicațiilor sau în seama dispozitivelor de comutare (routere și switch-uri).

Un set complet de cerințe fost elaborat pentru dispozitivele candidate la rețeaua TDAQ [6]. În paralel cu acest document, a fost definită și o metodologie de testare a echipamentelor. S-a efectuat un studiu de piață și în final au fost achiziționate dispozitivele care se încadrau cel mai bine în specificațiile ATLAS. În contextul acestui program de evaluare, a fost dezvoltat un sistem complet pentru testarea performanței și verificarea funcționalității dispozitivelor de rețea.

3 Testarea echipamentelor Ethernet

Un sistem de testare pentru rețele (*tester*) presupune transmiterea de trafic artificial către dispozitivul de pe bancul de test (*Device Under Test (DUT)*) și monitorizarea răspunsului, a “ieșirii” acestuia. Tester-ul trebuie să înregistreze orice schimbare a caracteristicilor fluxului de intrare la trecerea sa prin DUT: pierderile de pachete și întârzierile trebuie să fie cuantificate.

Pentru ATLAS era necesar un tester care să funcționeze la 1 Gbit/s și care să ofere cât mai multă flexibilitate în definirea tiparelor de trafic. Se dorea un ansamblu capabil să emuleze într-un mod realist aplicațiile și traficul din rețeaua TDAQ. Deoarece soluțiile bazate pe microprocesoare nu ar fi fost suficient de rapide, în final s-a optat pentru dezvoltarea unui sistem nou, bazat pe FPGA-uri.

3.1 Platforma GETB

Acesta a fost denumit *GETB, the Gigabit Ethernet Testbed* [7]. Platforma GETB are la bază o placă PCI (Figura 3.1) care conține un FPGA pe post de unitate centrală de procesare. Îi vom face o descriere succintă în cele ce urmează (detalii se regăsesc în Secțiunea 3.2 din lucrare).

3.1.1 Proiectarea și alegerea componentelor

La proiectul GETB, s-a ținut cont în primul rând de cerințele referitoare la performanță. De asemenea, s-a luat în considerare și faptul că partea hardware urma să fie produsă în serie mică și folosită într-un institut de cercetare. Schema electrică trebuia să fie cât mai simplă, cu un număr redus de componente. În acest fel se reducea posibilitatea apariției defectelor în procesul de fabricație.

Formatul PCI Am ales acest tip de interfață deoarece este compatibilă cu majoritatea calculatoarelor personale. De asemenea, un sistem cu granularitate mică oferă flexibilitate sporită în utilizare. Astfel am decis să construim o placă PCI cu numai două porturi Ethernet.

Unitatea de procesare Pentru unitatea centrală am avut de optat între un microprocesor de uz general, un FPGA sau un circuit integrat specializat (ASIC). Un ASIC oferă maximul posibil de

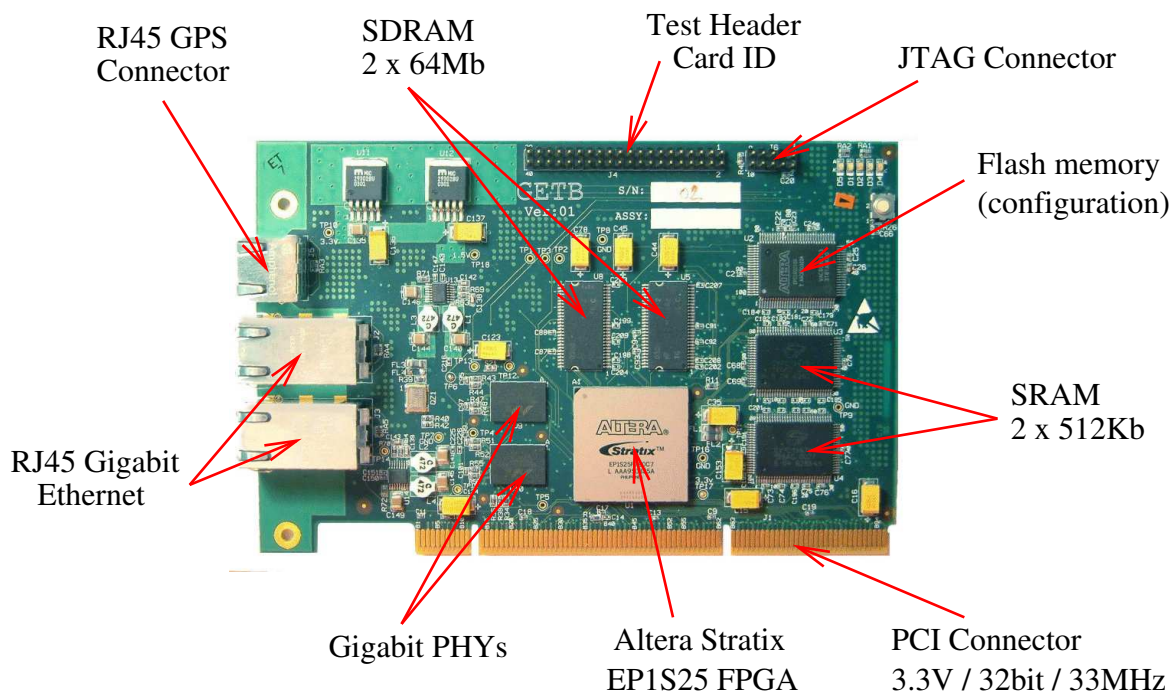


Figura 3.1: Placa PCI pentru platforma GETB.

viteză, dar costurile de dezvoltare sunt foarte mari. Un procesor de uz general este cel mai ușor de folosit (programat), dar nu se pretează la aplicații de timp real. FPGA-urile reprezintă o soluție de compromis – ciclul de dezvoltare este mult mai rapid decât la ASIC-uri și performanța este comparabilă. Un FPGA devine o alegere bună când putem beneficia de paralelismul masiv posibil pe acest tip de arhitectură. Pentru GETB a fost selectat un circuit de tip *Altera Stratix* conținând 25000 de blocuri logice. Comportamentul unui FPGA este definit de circuitul configurat în matricea de blocuri logice și conexiuni programabile. În continuare vom folosi denumirea de *firmware* pentru acest circuit.

Interfețele PCI și Ethernet După cum am spus la început, în timpul proiectării am încercat să reducem numărul de componente existente pe placa GETB. Astfel s-a decis integrarea circuitelor de control pentru PCI și Ethernet în interiorul FPGA-ului. În locul unor circuite dedicate (ASIC-uri), au fost achiziționate nuclee VHDL (*Intellectual Property Core, IP Core*) care aveau aceeași funcționalitate. Am folosit produsele *PLDA PCI Core* și *MoreThanIP Gigabit Ethernet MAC Core*. Cele două nuclee urmau să fie “legate” de restul firmware-ului, precum și de pinii de intrare/ieșire ai FPGA-ului – nucleul PCI la conectorul corespunzător și nucleele Ethernet la circuitele PHY asociate¹. Placa GETB conține două porturi Ethernet pentru transmisie prin cabluri de cupru (conectori RJ45, cabluri UTP). Ambele porturi sunt gestionate de același FPGA².

Memoria GETB-ul conține două tipuri de memorii. Există 128 Mbyte de memorie de tip SDRAM, cu o capacitate mare de stocare, optimizată pentru accesul la blocuri mari de date (transferuri în rafală). Memoria SDRAM păstrează o descriere a traficului ce urmează a fi transmis. De asemenea, în SDRAM se pot reține (temporar) pachete Ethernet. Pentru operațiile care necesită

¹PHY-ul este un circuit integrat care face legătura cu nivelul fizic Ethernet (codarea și decodarea semnalului de date).

²Al treilea conector RJ45 care apare în figură a fost prevăzut pentru a fi conectat la un sistem de distribuție a unui semnal de ceas global.

un acces rapid la date, am prevăzut 1 Mbyte de memorie de tip SRAM. Aceasta este folosită în principal pentru a crea histograme (care implică numeroase operații de tip “read-modify-write”).

Elemente pentru diagnostic Pentru a facilita dezvoltarea firmware-ului, placa mai dispune de LED-uri pentru diagnosticare și de un conector unde putem lega un analizor logic.

Configurarea FPGA-ului Sistemul GETB folosește un FPGA care stochează configurația în celule SRAM. Pentru inițializare, placa a fost prevăzută cu o memorie Flash. În timpul rulării, FPGA-ul poate fi reconfigurat prin intermediul conectorului JTAG (folosind un cablu de tip *Altera ByteBlaster*).

Schema electrică a plăcii GETB a fost realizată folosind mediul OrCAD. Placa a fost produsă de o firmă din Israel. Au fost construite 64 de astfel de plăci și toate cele 128 de porturi Ethernet au funcționat conform așteptărilor.

3.1.2 Mediul de dezvoltare

Descriem în cele ce urmează principalele unelte folosite la dezvoltarea firmware-ului pentru platforma GETB.

Handel-C Pentru codul utilizator s-a folosit limbajul Handel-C produs de compania Celoxica. Acesta este un limbaj pentru descriere hardware, asemănător din punct de vedere funcțional cu VHDL sau cu Verilog. Diferența fundamentală constă în faptul că sintaxa limbajului Handel-C este foarte asemănătoare cu cea a C-ului clasic. În plus, Handel-C conține câteva elemente specifice dezvoltării de hardware (circuite digitale). De exemplu, utilizatorul poate specifica blocuri de cod sursă care se vor executa în paralel – compilatorul va genera circuite logice separate pentru secvențele respective de cod. De asemenea, se mai pot specifica lățimile în biți pentru numerele întregi, există primitive pentru sincronizarea “proceselor” paralele și există posibilitatea creării de interfețe cu dispozitive sau nuclee VHDL externe (porturi de intrare / ieșire). Rezultatul compilării unui “program” Handel-C este descrierea unui circuit digital. Pentru a fi folosit într-un FPGA, acest circuit trebuie transformat astfel încât să fie implementat numai cu blocuri logice și conexiuni programabile. Aceasta sarcină revine uneltelor specifice Altera.

Altera Quartus II Acesta este numele unei suite de unelte software pentru compilarea circuitelor pentru FPGA-uri Altera. Programele din Quartus fac trecerea prin toți pașii necesari: sinteza circuitului pornind de la descrierea de nivel înalt (de tip VHDL), faza de transformare în blocuri logice (*technology mapping*) și, în final, faza cea mai dificilă, aceea de amplasare a blocurilor logice în matricea FPGA-ului și de alegere a traseelor între aceste blocuri (prin intermediul conexiunilor programabile). La terminarea acestor etape, Quartus produce un fișier binar care conține configurația FPGA-ului.

VHDL.Gen Circuitul programat într-un FPGA este alcătuit în general din mai multe blocuri funcționale. Acestea trebuie interconectate între ele iar o parte din porturile lor trebuie legate la pini de intrare/ieșire ai FPGA-ului. Aceste conexiuni pot fi realizate folosind o interfață grafică sau pot fi definite în limbajul VHDL. Ambele metode devin dificile și sunt predispuse la erori atunci când numărul de conexiuni este mare. În cazul platformei GETB aveam de realizat legături logice între 24 de blocuri funcționale, având în total 915 porturi. Pentru a simplifica această operațiune am creat o unealtă software denumită VHDL.Gen. Aceasta facilitează crearea legăturilor prin faptul că necesită la intrare doar o listă de reguli referitoare la modul în

care blocurile trebuiesc legate (minimul de informație necesar). Programul prelucrează această listă și generează un fișier VHDL care conține toate legăturile cerute. În plus, VHDL_Gen face o serie de verificări asupra regulilor inițiale și avertizează utilizatorul dacă acestea ar putea produce un circuit nefuncțional.

ModelSim Pentru simularea anumitor părți din firmware am folosit programul ModelSim. Acesta ne-a ajutat în special la început, înainte de a primi primele plăci GETB. Astfel am putut pregăti în prealabil câteva versiuni speciale de firmware pentru testarea anumitor componente de pe placă: memoriile, interfețele Ethernet, legătura prin PCI, etc.

Pentru a ține cont de dependențele între ele, programele descrise mai sus au fost integrate folosind utilitarul “make” disponibil sub sistemul de operare Linux. La început, pentru programarea FPGA-ului am folosit conectorul JTAG existent pe fiecare placă. Schimbarea firmware-ului pe toate cele 64 de plăci se face în prezent prin intermediul interfeței PCI – astfel putem transfera conținutul fișierului generat de Quartus direct în memoriile Flash.

3.1.3 Organizarea firmware-ului

Firmware-ul pentru placa GETB a fost proiectat în așa fel încât placa să poată fi folosită ca o platformă de dezvoltare. Codul sursă a fost astfel structurat încât să ofere o funcționalitate de bază și să fie ușor de extins. Această abordare a făcut posibilă folosirea plăcii în cadrul a trei proiecte diferite. Vom descrie în continuare structura internă a acestui firmware. O schemă bloc a acestuia se regăsește în Figura 3.2. Există următoarele două categorii de blocuri funcționale:

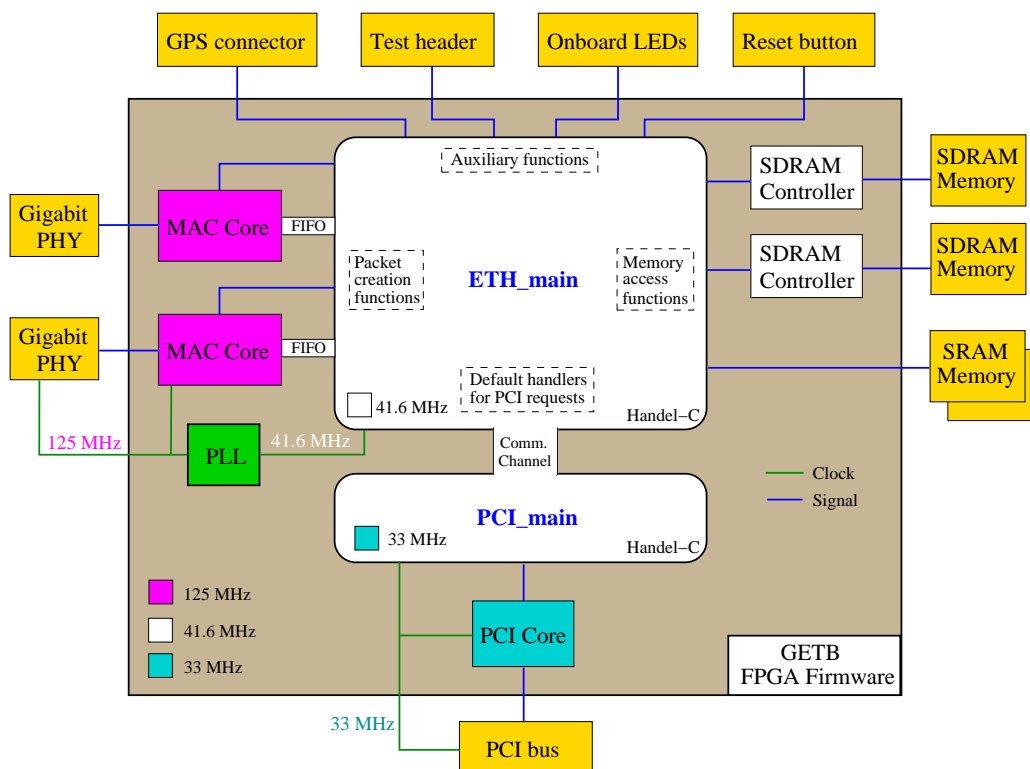


Figura 3.2: Structura firmware-ului din FPGA.

Interfețele cu exteriorul În această categorie intră nucleele de PCI și MAC (Ethernet), precum și controlerile pentru memoria SDRAM. Comunicația pe PCI se face la frecvența de 33 MHz.

Blocul logic	Porturi interne	Porturi externe (pini)	Gradul de ocupare
Nucleul PCI	14	23	4.3%
Controlerele de SDRAM (2x)	16	18	2.7%
Nucleele Ethernet MAC (2x)	108	20	32.1%
Elemente de infrastructură	50	26	0.1%
Codul utilizator (Handel-C)	402	97	54.5%
Întregul firmware	-	184 (471 pini)	93%

Tabelul 3.1: Blocurile logice din FPGA.

Transmisia și recepția datelor în standardul Gigabit Ethernet necesită o frecvență de ceas de 125 MHz. Codul aplicație (Handel-C) și memoriile funcționează la 41.6 MHz. Transferul datelor între aceste domenii de ceas se face cu ajutorul unor circuite de sincronizare instanțiate din Handel-C, precum și cu memorii SRAM de tip “dual-port”³.

Codul utilizator (Handel-C) Este alcătuit din două parti. Prima, numită PCI_main, interceptează cererile care vin dispre PCI și le transformă în comenzi pentru partea de cod care are acces direct la toate resursele hardware (ETH_main). PCI_main asigură și transferul rezultatelor înapoi către calculatorul gazdă (pentru citirile din memorii de exemplu). Cel mai important bloc de cod se numește ETH_main; acesta conține rutine de bază care facilitează accesul la restul componentelor de pe placă (memorii, Ethernet, LED-uri, etc). Blocul ETH_main înglobează un set de servicii de bază care să poată fi ușor extinse sau folosite pentru a crea diferite aplicații.

Se poate observa că o parte semnificativă din firmware este alocată interfețelor cu exteriorul (PCI, Ethernet, SDRAM). Pentru versiunea finală a sistemul de testare pentru rețele, gradul de ocupare al FPGA-ului era de 93%. Din Tabelul 3.1 putem vedea că mai mult de jumătate din matricea logică este ocupată de aplicația scrisă în Handel-C.

3.1.4 Optimizări

În cazul platformei GETB, FPGA-ul determină funcționalitatea întregului sistem. Deoarece nucleele care gestionau componente externe nu puteau fi eliminate sau modificate, codul utilizator a trebuit intens optimizat astfel încât să consume cât mai puține resurse logice și, în același timp, să satisfacă cerințele de viteză. O constrângere suplimentară a fost necesitatea de a controla cu un singur FPGA două porturi Ethernet. În cazul sistemului de testare, cele două porturi trebuiau să ofere același set de caracteristici și același nivel de performanță.

Pentru a depăși aceste obstacole, am folosit diferite tehnici de optimizare la nivelul codului utilizator scris în Handel-C. O listă completă a acestora se găsește în lucrare (Secțiunea 3.2.3.3). Enumerăm câteva dintre acestea în cele ce urmează.

Exploatarea paralelismului Transmisia și recepția pachetelor este realizată de blocuri care funcționează în paralel. De asemenea, în interiorul fiecărui bloc am folosit intensiv tehnica de “pipelining” pentru operațiile complexe care puteau fi descompuse în mai multe etape mai simple. Execuția paralelă și construirea de “pipeline”-uri (“linii de asamblare”) fac posibilă mărirea frecvenței de ceas.

Folosirea memoriilor în locul registrelor FPGA-urile moderne conțin celule de memorie SRAM, memorie foarte rapidă și care poate fi accesată ușor din Handel-C. Acolo unde a fost posibil,

³Memoriile dual-port sunt module SRAM integrate în FPGA. Ele dispun de două magistrale de date și două de adrese. Acestea pot fi accesate simultan. Cele două magistrale pot lucra la frecvențe de ceas diferite.

am înlocuit registrele cu memorii de acest tip. Această tehnică poate reduce mult ocuparea blocurilor logice din FPGA (care ar fi fost altfel folosite pentru registre).

Reducerea complexității secvențelor de cod combinatorial Expresiile de tip combinatorial (comparații, operații aritmetice) determină întârzierile din circuit și, implicit, frecvența maximă de operare. Am încercat să simplificăm pe cât posibil aceste expresii, uneori cu prețul unui grad de ocupare mai ridicat (s-a făcut un compromis între viteză și spațiul ocupat în matricea logică a FPGA-ului).

Aceste optimizări au făcut posibilă îndeplinirea celei mai importante cerințe inițiale: funcționarea plăcii cu ambele porturi operând la viteza de 1 Gbit/s în regim “full-duplex” (bi-direcțional) și pentru orice dimensiune de pachete.

3.1.5 Infrastructura de control

Sistemul de control pentru platforma GETB se bazează aproape în întregime pe limbajul de programare Python (un limbaj interpretat). Infrastructura de control a fost concepută pentru a fi folosită de toate proiectele bazate pe GETB. Ea este alcătuită din programe care asigură comunicația cu plăcile GETB (local sau prin rețea). Aceste programe sunt folosite pentru configurarea și monitorizarea plăcilor.

Sistemul de control al GETB-ului este alcătuit din două componente principale: un modul “server” care rulează pe calculatoarele care conțin plăci GETB și un modul “client” care asigură interfața cu utilizatorul (a se vedea și Figura 3.3).

Server-ul interacționează cu plăcile prin intermediul unui *driver* pentru sistemul de operare Linux. Serverele GETB fac publice un set de funcții de bază care pot fi folosite pentru controlul plăcilor de la distanță (din rețea). Aceste funcții pot fi apelate cu ajutorul unui mecanism RPC (Remote Procedure Call).

Clientul GETB este o componentă software care poate rula pe orice calculator unde există un interpretor pentru limbajul Python. Accesul la resursele oferite de serverele GETB este mai simplu prin intermediul clientului. Acesta se conectează prin rețea la toate serverele disponibile (la toate plăcile) și permite utilizatorului controlul oricărui port Ethernet de pe oricare dintre plăci (în mod transparent).

Prin intermediul clientului, utilizatorul are acces la o interfață în linie de comandă, poate rula script-uri (mici programe scrise în Python) și poate monitoriza toate plăcile folosind o interfață grafică.

La momentul actual există trei proiecte care folosesc platforma GETB: tester-ul pentru dispozitive Ethernet, un emulator de rețea și o aplicație similară sub-sistemului ROS din ATLAS. Cele trei proiecte au în comun o mare parte din firmware, precum și software-ul de control, pentru administrarea plăcilor.

3.2 Sistemul de testare

Folosind 64 de carduri GETB, am construit un sistem pentru testarea switch-urilor și router-elor cu până la 128 de porturi. Tester-ul poate trimite și recepționa pachete la viteza de 1 Gbit/s (la rata maximă de 1.5 milioane de pachete pe secundă sau la debitul maxim⁴ de 125 Mbyte/s). Plăcile GETB sunt instalate în calculatoare industriale (servere); controlul lor este centralizat (Figura 3.3).

La tester-ul GETB, transmisia are două moduri de funcționare. În primul, porturile Ethernet din tester sunt complet independente. În memoria plăcii se încarcă un set de descriptori. Aceștia sunt citați într-o

⁴Rata maximă este atinsă pentru pachetele mici, de 64 bytes. Debitul maxim se obține transferând pachete cu dimensiunea de 1518 bytes (maximul permis de Ethernet).

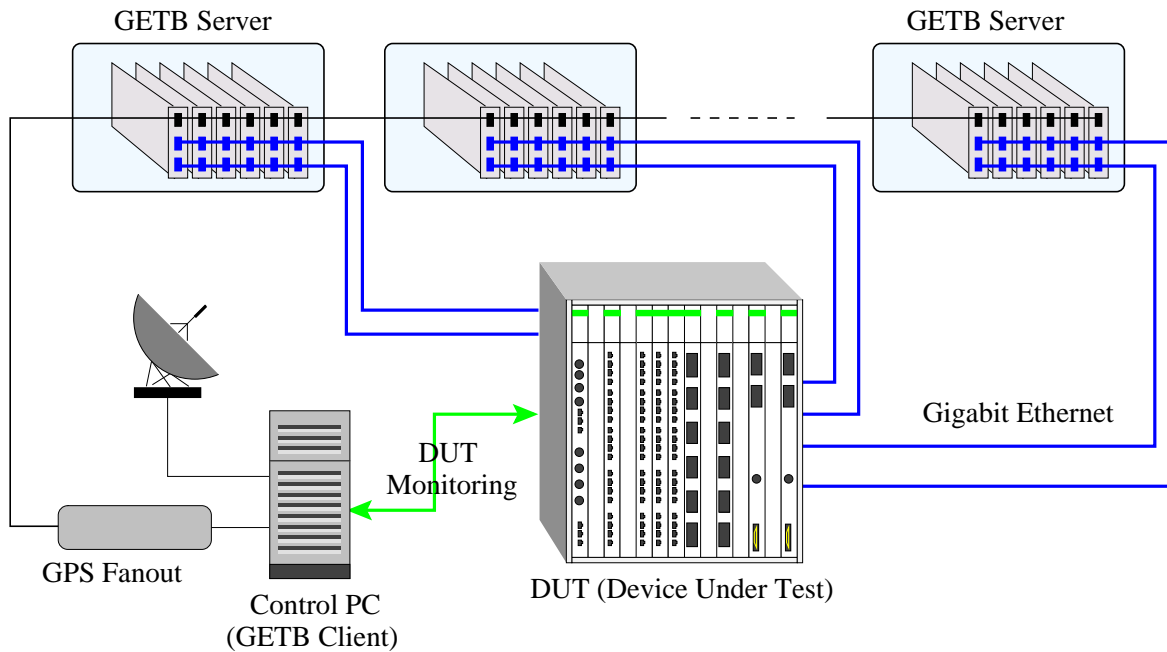


Figura 3.3: Diagramă cu tester-ul GETB conectat la un switch.

bucă și pornind de la fiecare dintre ei se generează câte un pachet. Într-un descriptor se specifică (cel puțin) dimensiunea pachetului, destinația și timpul de așteptare până la următorul pachet.

Al doilea mod de funcționare emulează traficul de tip cerere-răspuns din ATLAS. Există porturi care emit pachete conținând cereri către alte porturi care sunt capabile să genereze răspunsuri (clienți și servere). Clienții pot controla rata la care primesc răspunsurile limitând numărul de cereri trimise la un moment dat (detalii în Capitolul 4).

Tester-ul GETB menține statistici cu privire la numărul de pachete trimise și recepționate. Există și posibilitatea de a face histogramme pentru anumite câmpuri extrase din pachetele primite (de exemplu, pentru timpul dintre două pachete consecutive). De asemenea, sistemul GETB permite măsurarea precisă a întârzierilor prin rețea (latența). Opțional, la recepție poate fi activată și captura de pachete. Astfel se poate face o analiză amănunțită a traficului, având informații despre toate pachetele recepționate. Datele capturate la recepție pot fi corelate în timp deoarece ceasurile interne ale plăcilor sunt sincronizate.

Pentru tester-ul GETB, am dezvoltat o bibliotecă de funcții Python care conține implementări pentru toate testele definite în metodologia standard pentru testarea dispozitivelor din rețeaua ATLAS [6]. Sistemul produce grafice în mod automat pornind de la măsurători. Consistența rezultatelor este de asemenea verificată – în multe cazuri întreaga suită de teste poate fi rulată fără intervenția utilizatorului.

Pentru exemplificare, prezentăm în Figura 3.4 un rezultat tipic obținut folosind tester-ul GETB. Graficul arată cum crește latența printr-un switch, pe măsură ce crește gradul de încărcare la intrare. La switch au fost conectate $N = 48$ porturi din tester. Fiecare port trimitea pachete la toate celelalte $N - 1$ destinații. Acest tip de trafic se numește “complet întrețesut” (*full-mesh*). La recepție s-a înregistrat latența medie a pachetelor primite. Atunci când crește rata la toate porturile de intrare, crește și ocuparea medie a cozilor din interiorul switch-ului (la ieșire) – aceasta duce, în mod implicit, la o creștere a întârzierilor, fenomen ce poate fi observat în figură.

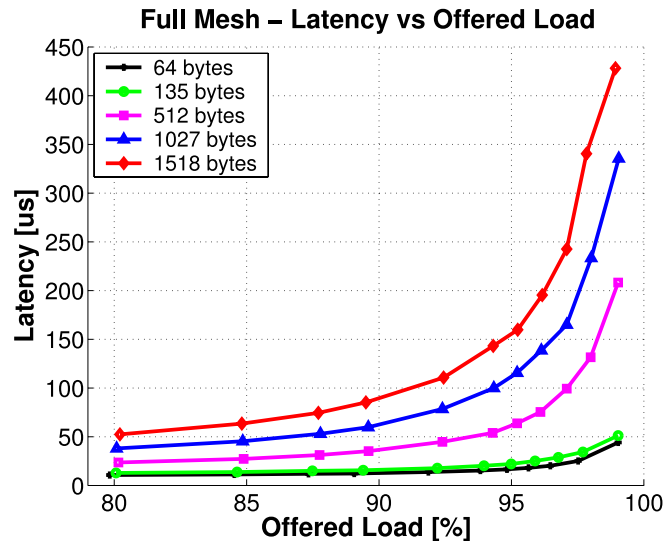


Figura 3.4: Latența în funcție de încărcare (trafic complet întrețesut, *full-mesh*).

4 Analiza traficului din rețeaua TDAQ

Din punct de vedere al utilizării rețelei, aplicațiile ATLAS se împart în servere și clienți. Clienții trimit cereri către servere și așteaptă să primească răspunsurile cu date despre evenimente. Pachetele care conțin cereri au dimensiunea de 64 bytes (minimum pentru Ethernet), în timp ce răspunsurile au 1518 bytes (raport de 1:25). Din acest motiv, clientul poate foarte ușor să trimită în succesiune rapidă prea multe cereri. Într-un astfel de caz, va primi o avalanșă de răspunsuri care nu vor putea fi procesate suficient de repede. De asemenea, este posibil ca fluxurile de date cu răspunsurile să creeze congestie în rețea – aceasta va duce la pierderea unor pachete răspuns.

Pentru a preveni astfel de situații și pentru a păstra un control asupra cantității de date care circulă prin rețea, clienții limitează numărul de cereri trimise și pentru care se așteaptă un răspuns. Se folosește un mecanism bazat pe jetoane (*token*) pentru ajustarea fluxului de date (*traffic shaping*). Clientul asociază un jeton fiecărei cereri trimise. Există un număr maxim de jetoane pe care le poate deține clientul la un moment dat: *High Watermark*, H_W . Clientul trimite cereri până atinge H_W , limita superioară. După care se oprește și așteaptă să primească răspunsurile, pierzând câte un jeton pentru fiecare răspuns primit. Când numărul de jetoane (*Token Counter*, T_C) scade sub o limită inferioară (*Low Watermark*, L_W), clientul reia transmisia de cereri, trimițându-le în rafală până când ajunge din nou la H_W .

Mecanismul bazat pe jetoane trebuie adaptat la caracteristicile rețelei. Viteza la care clientul primește răspunsuri este determinată de limitele H_W și L_W , precum și de întârzierile din rețea.

Am folosit tester-ul GETB pentru a studia traficul produs în rețea de mecanismul descris mai sus. GETB-ul poate fi folosit pentru a emula atât “serverele”, cât și “clienții” care trimit cereri în funcție de numărul de jetoane disponibile. Activând modul în care se salvează informații pentru fiecare pachet primit, am putut vedea exact cum variază numărul de jetoane la client, precum și momentele la care sunt primite cererile și răspunsurile.

4.1 Viteza la recepție

Folosind informațiile capturate de plăcile GETB, am determinat funcția care descrie rata la care clientul primește răspunsurile. Demonstrația a pornit de la definiția noțiunii de viteză: $\frac{\text{Cantitate de date}}{\text{Timp}}$. Comportamentul clientului este ciclic datorită mecanismului care reglează traficul, așa dar analiza s-a

făcut pentru o singură perioadă.

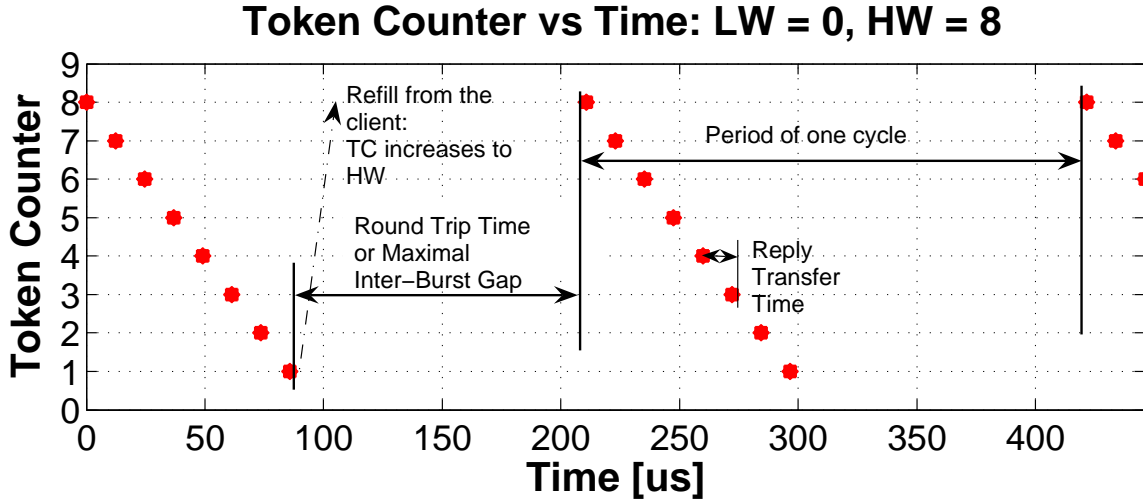


Figura 4.1: Variația în timp a numărului de jetoane (la un client cu $L_W = 0$, $H_W = 8$).

În Figura 4.1 putem vedea cum variază numărul de jetoane pentru $H_W = 8$ și $L_W = 0$. Înregistrarea a fost făcută la client, din momentul în care s-au primit primele răspunsuri – acesta este motivul pentru care $T_C = 8$ la început¹. Fiecare punct din figură reprezintă un răspuns primit de client. Valoarea lui T_C scade pe măsură ce ajung răspunsuri de la server. Când $T_C = L_W$, clientul trimite un nou set de cereri. Prima ajunge la server după D_{req} secunde (*request delay*). Următoarele se succedă într-o scurtă rafală. Primul răspuns ajunge înapoi la client după alte D_{rpl} secunde (latența răspunsului prin rețea, *reply delay*). Astfel se explică pauza care se poate observa în figură. Notăm durata de timp $D_{req} + D_{rpl}$ cu T_0 . După T_0 secunde, clientul începe să primească răspunsurile. Acestea sosesc la intervale de T_{rpl} secunde (timpul necesar transferului din rețea²). După 8 răspunsuri, T_C ajunge la $L_W = 0$ și un nou ciclu poate începe: clientul emite iar 8 cereri și datele sosesc după alte T_0 secunde.

Viteza de recepție la client este dată de: 1) distanța în timp între două “rafale” cu răspunsuri (*Inter-Burst Gap*, IBG^3) și 2) de lungimea unor astfel de rafale (*Burst Length*, BL). În lucrare, în Capitolul 4 arătăm că IBG scade pe măsură ce crește limita L_W , în timp ce BL depinde de diferența $H_W - L_W$. În final se ajunge la următoarea funcție pentru viteza cu care se primesc răspunsurile la client:

$$R_x(L_W, H_W; S_{rpl}, T_{rpl}, T_0) = \frac{\overbrace{(H_W - L_W) \cdot \left\lfloor \frac{H_W}{H_W - L_W} \right\rfloor \cdot S_{rpl}}^{\text{Burst Length}}}{\underbrace{R(T_0 - L_W \cdot T_{rpl})}_{\text{Inter Burst Gap}} + (H_W - L_W) \cdot \left\lfloor \frac{H_W}{H_W - L_W} \right\rfloor \cdot T_{rpl}} \quad (4.1)$$

Expresia⁴ depinde de dimensiunea răspunsurilor (S_{rpl} , *reply size*), dar nu și de cea a cererilor deoarece s-a considerat că acestea au dimensiuni neglijabile în comparație cu răspunsurile.

¹Numărul de jetoane ajunge la $H_W = 8$ după ce se trimit toate cererile.

²Durata T_{rpl} depinde de viteza de transfer a rețelei și de dimensiunea pachetului. Pentru pachete de 1518 bytes transferate prin Gigabit Ethernet avem $T_{rpl} = 12.3\mu s$.

³Pentru exemplul din Figura 4.1 avem $IBG = T_0$.

⁴Am notat cu $R(x)$ funcția $R(x \geq 0) = x$ și $R(x < 0) = 0$.

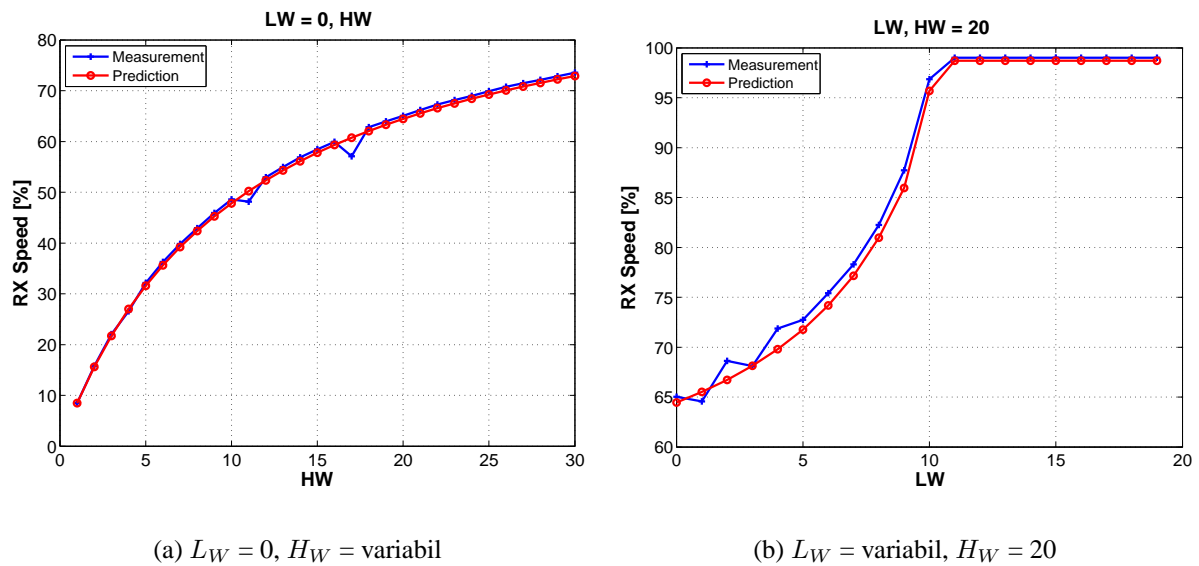


Figura 4.2: Viteza la recepție – Măsurători și curbe teoretice.

Măsurătorile au arătat că valorile prezise de formula (4.1) se potrivesc foarte bine cu rezultatele experimentale. Un exemplu este ilustrat în Figura 4.2. S-a măsurat rata de recepție la client în două situații: pentru L_W constant și pentru H_W constant. În ambele cazuri cele două curbe, cea teoretică și cea experimentală, se suprapun aproape în totalitate. În lucrare a fost analizată întreaga suprafață 3D descrisă de funcția $Rx(L_W, H_W)$ (4.1). S-a constatat că eroarea relativă dintre valorile prezise și măsurători este în medie mai mică de 1%.

4.2 Gradul de ocupare al coziilor

O caracteristică a traficului de tip cerere-răspuns este aceea că în orice moment, o cantitate mare de date poate fi direcționată către o anumită (unică) destinație din rețea – destinația fiind unul dintre clienții care emit cereri. Timpul necesar unui client pentru a trimite un set de cereri este foarte mic (câteva microsecunde). Astfel, cererile vor ajunge practic aproape în același timp, atunci când sunt trimise către servere diferite. Acestea vor începe să răspundă la viteza maximă posibilă (1 Gbit/s).

Presupunând că serverele și clientul sunt conectate la același switch, atunci vom observa o acumulare de date la portul de ieșire dinspre switch către client. Switch-ul va trebui să rețină într-o memorie tampon aproape toate pachetele care vin de la servere, deoarece nu poate direcționa mai multe fluxuri simultane de 1 Gbit/s către o linie de ieșire cu capacitatea maximă tot de 1 Gbit/s.

Când memoria tampon (*buffer*) este plină, switch-ul începe să piardă pachete. Folosind sistemul GETB, am studiat dependența dintre gradul de ocupare al buffer-elor din switch și modul în care este configurat clientul. În lucrare, în Secțiunea 4.6, sunt deduse expresiile care descriu nivelurile de utilizare minime, medii și maxime.

Valorile minime depind de limita inferioară, L_W . Regulile de funcționare ale clientului spun că la orice moment trebuie să existe cel puțin L_W răspunsuri în așteptare. În mod similar, H_W determină numărul maxim de răspunsuri care circulă prin rețea, deci ocuparea maximă a coziilor va fi proporțională cu H_W . Formulele care apar în lucrare țin seamă și de faptul că putem avea și cereri sau răspunsuri care sunt în curs de propagare prin rețea. Când timpii de propagare sunt mari, putem avea multe pachete în

tranzit. Pentru a ține cont de acest efect, am definit “capacitatea rețelei” (N_C) ca fiind egală cu numărul de răspunsuri care pot fi în tranzit la un moment dat.

Pentru ocuparea medie, a fost nevoie să analizăm variația în timp a gradului de ocupare al buffer-ului. Pentru latențe mici în rețea, ocuparea medie este aproximativ proporțională cu media aritmetică a celor două limite (L_W, H_W) – ceea ce este intuitiv dacă presupunem că ocuparea variază în mod uniform între minim și maxim. Expresia la care ajungem în lucrare descrie bine și variația cozilor atunci când întârzierile din rețea sunt mai mari. În astfel de cazuri s-a ținut cont de viteza de umplere și de golire a buffer-ului din switch, precum și de faptul că există intervale de timp când buffer-ul este gol.

4.3 Aplicații

Pe lângă tester-ul GETB, a fost dezvoltat și un set de programe care implementează în software mecanismul bazat pe jetoane folosit pentru controlul traficului. Acest set de programe (denumite mpNetPerf) pot fi instalate pe toate calculatoarele din sistemul TDAQ. Ele ne permit să testăm întreaga rețea pentru a vedea cum suportă traficul de tip client-server.

De asemenea, este posibil să măsurăm dimensiunea buffer-ului la ieșirea unui switch, dacă ținem cont de relația dintre limitele (L_W, H_W) și ocuparea acestui buffer. Atunci când $L_W = H_W - 1$, cantitatea medie de date care așteaptă în buffer-ul către client este de aproximativ $H_W \cdot S_{rpl}$ bytes. Dacă H_W este prea mare, clientul va pierde răspunsuri. Dimensiunea buffer-ului este dată de cea mai mare valoare a lui H_W pentru care nu se înregistrează pierderi.

5 Unelte pentru gestiunea rețelelor

Instalarea efectivă a echipamentelor în rețeaua TDAQ a început în anul 2006. Până la ora actuală au fost montate 62 de switch-uri și routere. Sistemul final va conține aproape 300 de dispozitive de rețea și peste 3000 de calculatoare. Majoritatea calculatoarelor vor avea cel puțin două legături la rețea (una pentru control și una pentru date). Operatorul uman nu poate întreține un sistem de asemenea dimensiuni decât cu ajutorul unor unelte care să automatizeze o parte din sarcini. Rețeaua TDAQ va fi administrată folosind atât produse comerciale cât și sisteme construite la CERN și adaptate cerințelor specifice experimentului ATLAS. De exemplu, gestiunea echipamentelor și monitorizarea traficului se va face (cel puțin într-o primă fază) folosind programe dezvoltate la CERN. În [8] se face o descriere a planurilor actuale referitoare la modul de operare al rețelei TDAQ.

5.1 Configurarea dispozitivelor

Pentru a se integra într-o rețea existentă, orice dispozitiv trebuie mai întâi configurat. Din păcate, metoda prin care se dau valori parametrilor de configurare este specifică fiecărui fabricant. Deși a fost propus un standard în acest sens [9], puține dispozitive actuale îl implementează.

Folosind limbajul Python, am dezvoltat un modul (o bibliotecă de funcții) care permite utilizatorului să scrie mici programe (*script-uri*) pentru a configura echipamentele de rețea. Programele care folosesc acest modul (denumit *sw_script*) vor funcționa la fel pe orice tip de dispozitiv. Sistemul *sw_script* face o translație între apelurile de funcții din Python și metodele de configurare specifice fiecărui tip de switch sau router.

Acest modul a fost folosit în cadrul mai multor proiecte: de exemplu, sistemul GETB poate rula testele pe un anumit dispozitiv în mai multe configurații – acestea sunt schimbate în mod automat

fără să necesite intervenția utilizatorului. O altă aplicație a constatat într-un set de unelte care modifică configurația tuturor switch-urilor de un anumit tip. Programele care folosesc `sw_script` au acces la toate facilitățile oferite de limbajul Python (de exemplu programarea orientată pe obiecte). Dezvoltarea unor sisteme “semi-inteligente”, care sunt capabile să tolereze anumite erori, devine astfel mai ușoară. Dar cea mai importantă aplicație a modulului `sw_script` a fost implementarea unui sistem care facilitează ținerea unui inventar cu întreaga structură a rețelei. Prin “structură” înțelegem totalitatea legăturilor fizice existente în rețea.

5.2 Descoperirea topologiei rețelei

Cunoașterea poziției exacte a legăturilor fizice (a topologiei) este esențială pentru diagnosticarea problemelor și pentru păstrarea unei documentații la zi a rețelei. În general topologia văzută la nivelurile 1 (fizic¹) sau 2 (legătura de date) diferă de cea de nivel 3 (nivelul rețea). O hartă de nivel 3 este ușor de obținut deoarece orice router păstrează o listă cu router-ele din imediata sa vecinătate. Odată obținute aceste liste din toate router-ele, topologia de nivel 3 se poate deduce cu ușurință.

Problema devine mai complicată pentru legăturile fizice (topologia reală a rețelei), deoarece dispozitivele de nivel 2 (switch-urile) nu mențin informații despre switch-urile învecinate. Alte metode trebuie folosite în acest caz. O prezentare a acestora este disponibilă în [10].

Orice calculator sau switch care face parte dintr-o rețea Ethernet are asociat un identificator unic: adresa MAC, un număr pe 48 de biți. Switch-urile comută pachete în funcție de aceste adrese. Un switch învață ce adrese sunt accesibile prin fiecare interfață fizică (port). Procesul de învățare este iterativ și se bazează pe citirea adreselor sursă din pachetele primite.

Dacă pachetele de la o anumită adresă au sosit printr-un anumit port, atunci se presupune că pachetele către acea adresă vor trebui trimise prin același port. Această presupunere este validă numai dacă rețeaua nu conține bucle (este construită ca un arbore).

Folosind tabelele cu adrese MAC (*MAC Address Tables*) din fiecare switch din rețea, putem descoperi legăturile dintre switch-uri precum și porturile unde sunt legate calculatoarele (Figura 5.1). Figura din stânga ilustrează 5 switch-uri Ethernet (S1-5) cu tabelele MAC asociate fiecărui port. Se dorește determinarea conexiunilor între aceste dispozitive (figura din dreapta).

Metoda prezentată în lucrare pornește de la observația că dacă există o legătură directă între două switch-uri, atunci listele cu adrese MAC de la cele două capete (porturi) nu pot avea adrese comune. Să presupunem că o adresă MAC apare în ambele liste asociate porturilor de la capetele unei legături directe dintre două switch-uri. Când un pachet destinat acestei adrese va ajunge la unul din switch-uri, pachetul va intra într-o buclă, fiind transferat de la un switch la altul la infinit (prin acea legătură directă care am presupus că există). Identificarea unor elemente comune în listele de adrese ale celor două porturi, ne indică faptul că între cele două switch-uri mai există alte elemente de rețea. Când listele sunt disjuncte, trebuie să existe o legătură de nivel 2 între cele două switch-uri (în lucrare adăugăm două condiții suplimentare pentru a ne asigura că cele două switch-uri nu fac parte din rețele complet separate).

Lucrarea conține o demonstrație pentru afirmația din paragraful anterior și prezintă un algoritm care descoperă topologia unei rețele de nivel 2. Algoritmul este ulterior îmbunătățit pentru a descoperi și legăturile cu elemente de nivel 3 (routere). Arătăm cum pot fi recunoscute conexiunile între routere, precum și cele între routere și switch-uri.

Procedura de descoperire a topologiei pe baza tabelor cu adrese MAC a fost implementată folosind

¹Ne referim aici la nivelurile din stiva OSI (Open Systems Interconnect).

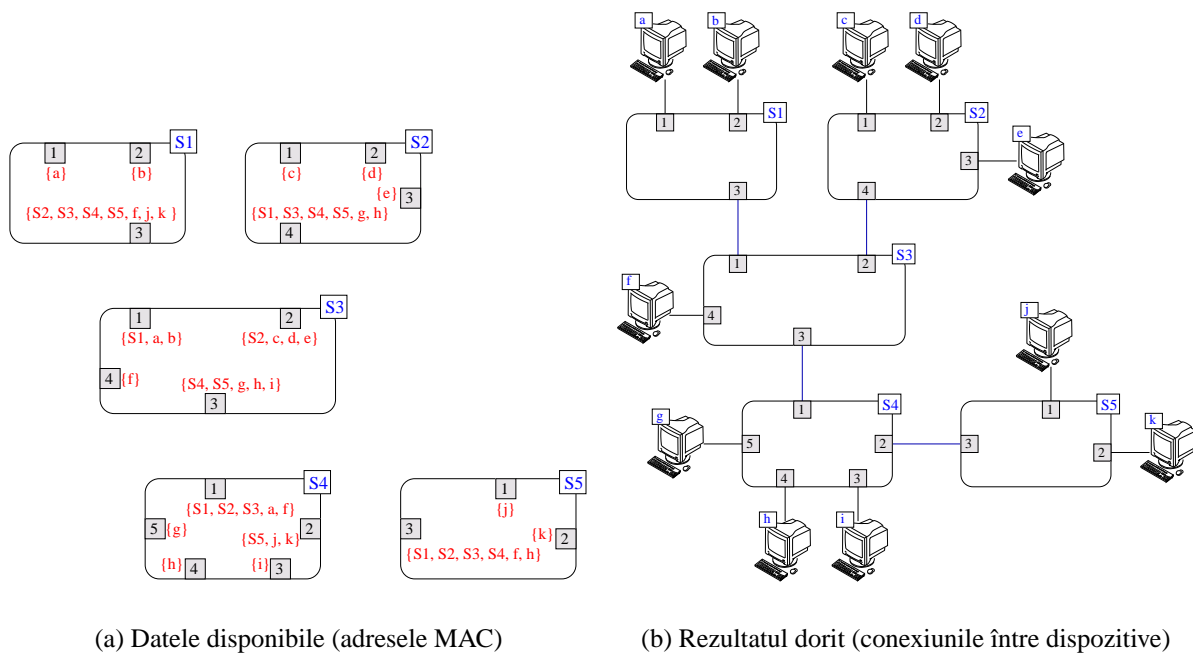


Figura 5.1: Problema descoperirii topologiei unei rețele.

modulul `sw_script` descris în Secțiunea 5.1. Pentru rețeaua ATLAS, am implementat un sistem care inventariază periodic toate echipamentele instalate. Acesta a fost folosit cu succes pentru a verifica toate conexiunile din rețea și pentru a le detecta pe cele incorecte.

Topologia rețelei este deosebit de utilă pentru vizualizarea traficului: a fluxurilor de date care circulă prin rețea. Cu ajutorul bibliotecii `sw_script` și al unei aplicații pentru reprezentarea grafurilor [11], am implementat un sistem de monitorizare a rețelei. Graful asociat întregii rețele este afișat, iar culoarea fiecărei legături se schimbă în funcție de gradul de utilizare. Acest sistem de monitorizare este actualizat în timp real și ne permite să aflăm imediat care sunt cele mai încărcate segmente de rețea.

6 Monitorizarea traficului prin eșantionare

Metodele clasice pentru monitorizarea traficului sunt bazate pe protocolul SNMP (*Simple Network Management Protocol*). SNMP ne permite să citim toate registrele care numără pachetele și dimensiunile acestora, pentru fiecare port al unui switch. Folosind aceste date, putem calcula, pentru orice interfață, lățimea de bandă utilizată (gradul de ocupare). Din păcate, SNMP-ul nu oferă informații despre tipul de trafic care circulă pe o anumită legătură. Pentru un singur port, soluția constă în instalarea unui analizor de rețea, un dispozitiv care permite inspectarea tuturor pachetelor. Un alt gen de abordare este necesară într-o rețea de câteva mii de porturi.

Un analizor de rețea are de prelucrat o cantitate enormă de date – 1.2 Gbyte/s pentru o linie de 10 Gbit/s care transportă numai pachete de 1518 bytes. Prin eșantionarea pachetelor, cantitatea de date poate fi redusă în mod substanțial. Dacă în loc să salveze fiecare pachet, analizorul păstrează doar un pachet din 4000, atunci cantitatea de date se reduce la 300 Kbyte/s. Pentru a analiza traficul, în general suntem interesați de antetele pachetelor (*header-e*). Încă un mod de a micșora volumul de date este de a salva doar primii bytes din fiecare pachet, cei care conțin antetul (în general 64 bytes). Astfel se ajunge la

aproximativ 13 Kbyte/s pentru o linie de 10 Gbit/s – un volum care poate fi ușor stocat sau procesat.

6.1 Valori estimative

Să presupunem că avem un analizor de rețea care eșantionează în medie n pachete dintr-un număr total de N . Cele n pachete salvate sunt clasificate și grupate în funcție de anumite proprietăți (de exemplu în funcție de perechiile de adrese sursă-destinație). Presupunem că avem c eșantioane într-una din aceste grupe (G). Se pune problema estimării dimensiunii totale a mulțimii G , dacă s-ar fi salvat toate cele N pachete.

Intuitiv, putem presupune că proporțiile se păstrează și că vom avea $\hat{N}_c = \frac{c}{n} \cdot N$ elemente din clasa G în setul total de N pachete. Se demonstrează că \hat{N}_c este un bun estimat al variabilei aleatoare N_c . Se poate calcula de asemenea și eroarea relativă a estimării, arătându-se că este proporțională cu $\sqrt{\frac{1}{c}}$ (pentru un număr mare de eșantioane). Folosind \hat{N}_c , lărgimea de bandă utilizată de pachetele (fluxul de date) din gruparea G poate fi estimată. Mai trebuie cunoscute doar dimensiunile pachetelor eșantionate, precum și cantitatea totală de date pentru cele N pachete.

Procesul descris mai sus ia în medie un eșantion la fiecare $\frac{N}{n}$ pachete. S-a observat că precizia este bună chiar și pentru valori mari ale acestui raport (de ordinul sutelor sau miilor). Pentru valori $\frac{N}{n}$ relativ mari, procesul de eșantionare are nevoie de puține resurse de calcul. Ținând cont de această observație, în anul 2001 a fost definit un standard care propune atașarea unui sistem de eșantionare la fiecare port al unui switch sau router.

6.2 Standardul sFlow

RFC 3176, cunoscut sub numele de *sFlow* [12] este numele standardului pentru eșantionarea pachetelor. Un sistem complet *sFlow* este alcătuit din două componente: un agent și un colector (Figura 6.1). Agentul *sFlow* este implementat în hardware în interiorul switch-ului sau router-ului. Responsabilitatea agentului este să eșantioneze pachetele la o rată dată și să trimită eșantioanele către colector. Colectorul este o aplicație software care rulează pe orice calculator din rețea. Eșantioane de la toate dispozitivele sunt primite de colector. Acesta le sortează și poate calcula valorile estimative pentru fiecare clasă de trafic (după cum s-a arătat în secțiunea anterioară).

Standardul *sFlow* este gândit astfel încât să necesite un minim de resurse hardware în echipamentul de rețea. Prelucrarea și arhivarea datelor se fac numai la colector. Eșantioanele sunt trimise sub formă de pachete UDP (mai multe eșantioane per pachet).

Un detaliu important este acela că eșantionarea nu se face cu o rată constantă. Agentul trebuie să captureze în medie un pachet din N (de la fiecare port). Numărul de pachete dintre două eșantioane este aleator, dar în medie trebuie să fie egal cu N (un parametru configurabil).

6.3 Exemplu de aplicație

Pentru a evalua posibilitățile standardului, am dezvoltat un sistem pentru colectarea și analiza eșantioanelor *sFlow*. Aplicația a fost scrisă în Python și testată pe câteva din switch-urile candidate la rețeaua TDAQ din ATLAS.

Eșantionul asociat unui pachet conține informațiile din antetele Ethernet și IP (adrese MAC, adrese IP) și câteva date legate de procesul de comutare (*switching*): portul de la care a fost primit pachetul și portul unde urmează să fie trimis.

Colectorul *sFlow* clasifică eșantioanele primite în funcție de switch, port și o “semnătură” sau “cheie”

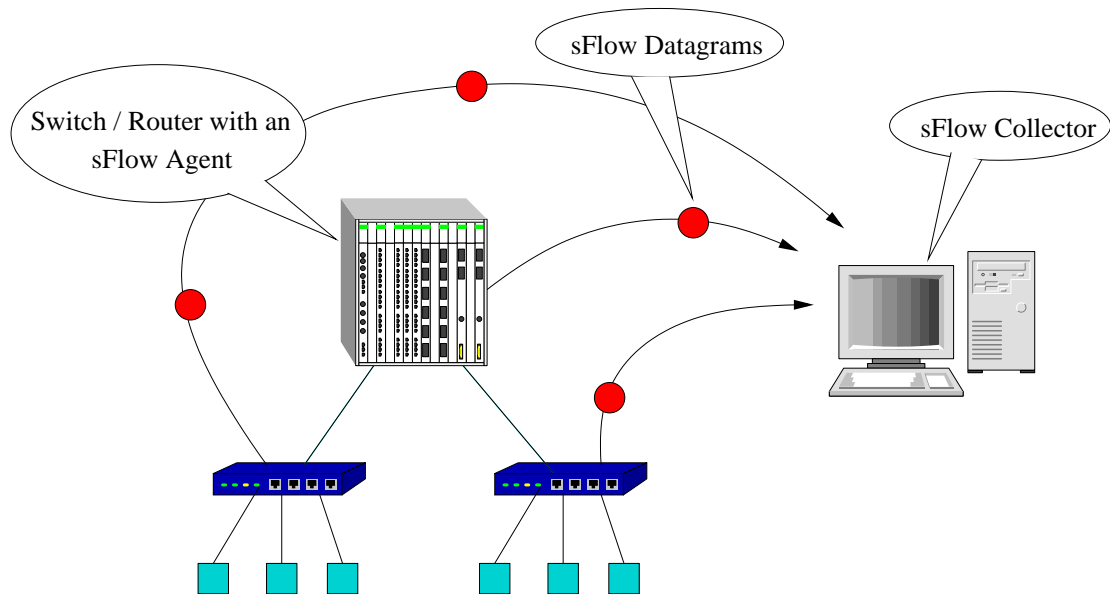


Figura 6.1: Agenții *sFlow* din dispozitive trimițând date la colector.

calculată pentru fiecare pachet. În general această “cheie” este alcătuită din adresele sursă și destinație. Astfel putem cataloga toate pachetele care fac parte din același flux de date (*flow*).

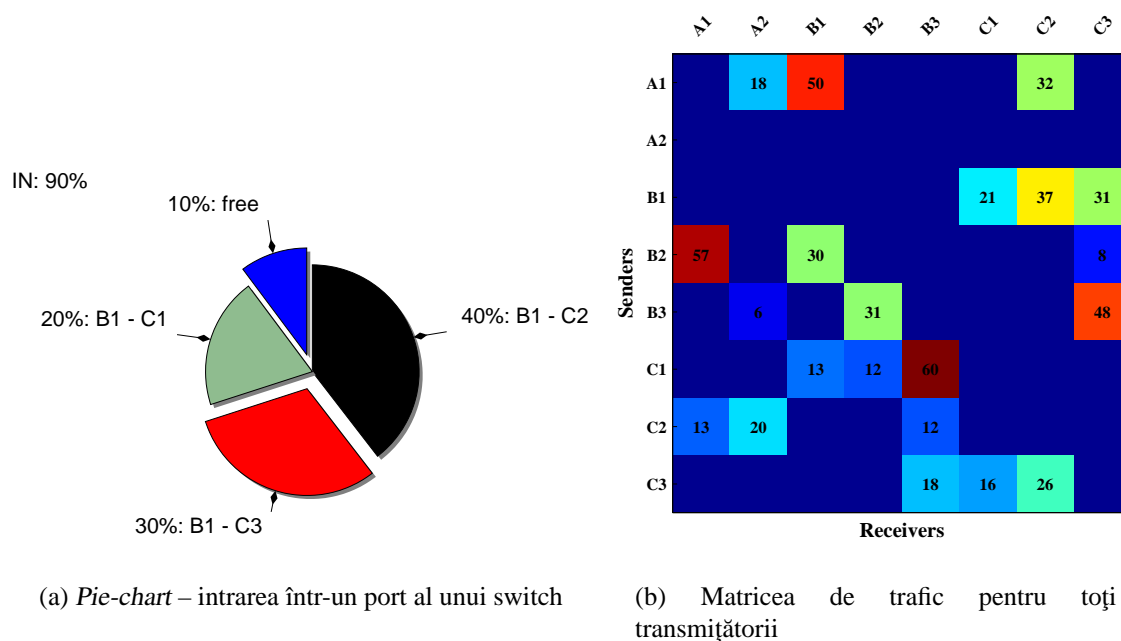
Pentru fiecare flux descoperit, colectorul estimează lărgimea de bandă utilizată. Acest calcul se face pentru fiecare port și pentru fiecare switch din rețea. Pentru o precizie mai bună, colectorul folosește eșantioanele recepționate în decursul ultimelor 4-5 minute.

Cu datele astfel obținute, se pot genera grafice cu cele mai “active” conversații din rețea. La nivelul unui port, se poate construi un grafic de tip *pie-chart* ca în Figura 6.2(a). Figura reprezintă o sursă $B1$ care trimite pachete către trei destinații: $C1, C2, C3$. Dacă am fi utilizat numai SNMP, am fi văzut că 90% din linie este ocupată. Cu *sFlow* putem afla cum este împărțită lărgimea de bandă între cele trei conversații. Adunând datele referitoare la transmisie de la toate porturile unui switch sau din toată rețeaua, aflăm lărgimea de bandă ocupată de fiecare conversație. Acest set de date poate fi reprezentat ca o matrice în care elementul (i, j) conține banda utilizată pentru transmisia de la nodul i la nodul j . Un exemplu apare în Figura 6.2(b). Cu acest tip de reprezentare, avem o viziune globală asupra întregii rețele.

Folosind colectorul construit și sistemul GETB am studiat și performanța unui agent *sFlow*. Am arătat în lucrare că pentru anumite combinații de fluxuri de pachete¹, agentul nu a putut eșantiona traficul la viteza cerută. În astfel de cazuri, estimările facute folosind tehnologia *sFlow* au erori foarte mari. Din fericire, echipamentele achiziționate pentru rețeaua TDAQ nu au astfel de probleme.

Tehnologia *sFlow* este deosebit de utilă atunci când trebuie diagnosticată congestia din rețea. Folosind un colector *sFlow* putem identifica rapid care sunt fluxurile cu cea mai mare contribuție la traficul dintr-un anumit punct al rețelei (în general o legătură “supra-încărcată” între două switch-uri).

¹Fluxuri alcătuite din pachete foarte mici amestecate cu pachete foarte mari.

Figura 6.2: Rezultate tipice obținute folosind *sFlow*.

7 Încheiere

Rețeaua TDAQ este o componentă vitală a sistemului de achiziție al experimentului ATLAS. Proiectarea ei a necesitat studii amănunțite și efortul susținut al unei întregi echipe de cercetare. Autorul a contribuit la evoluția și perfecționarea rețelei TDAQ în toate etapele ei de dezvoltare.

7.1 Contribuții

Proiectarea rețelei TDAQ Pe toată durata perioadei de proiectare, autorul a contribuit activ la elaborarea și îmbunătățirea arhitecturii rețelei ([3], [4]) – aceasta a fost finalizată în 2006 [5].

În urma unui studiu al tehnologiilor existente, s-a decis ca rețeaua TDAQ să fie bazată pe Ethernet. Pentru a selecta cele mai potrivite dispozitive, autorul a participat la definirea unui set de cerințe referitoare la nivelurile de performanță și la caracteristicile funcționale [6]. Aceste cerințe au fost însoțite de o metodologie completă de testare (verificare). Autorul a avut o contribuție majoră la dezvoltarea unei platforme hardware pentru aplicații în rețele Ethernet. Platforma GETB, care folosește FPGA-uri pentru procesarea datelor, a fost folosită la realizarea unui sistem de testare pentru switch-uri și routere Gigabit Ethernet (Capitolul 3 și articolul [7]). Cu ajutorul acestui sistem, am putut testa toate dispozitivele propuse pentru rețeaua TDAQ. Autorul a implementat firmware-ul pentru FPGA și întreaga infrastructură software pentru controlul platformei GETB. Pe lângă tester-ul pentru rețele, alte două proiecte au fost finalizate cu succes folosind platforma GETB.

Flexibilitatea sistemului GETB ne-a permis efectuarea unui studiu amănunțit al traficului generat de aplicațiile din sistemul de achiziție ATLAS. Autorul a analizat acest tipar de trafic și a determinat funcțiile care descriu lărgimea de bandă utilizată precum și gradul de ocupare al cozilor din rețea (Capitolul 4).

Instalarea și administrarea echipamentelor Autorul e dezvoltat un set de unelte software care sim-

plifică gestiunea dispozitivelor instalate în rețea. Pentru a automatiza procesul de configurare al switch-urilor și router-elor, a fost creată o bibliotecă de funcții (modul) în limbajul Python (Secțiunea 5.1). Cu ajutorul acestui modul se pot scrie programe care aduc modificări asupra configurației echipamentelor. Aceste programe nu necesită schimbări atunci când acționează asupra unor dispozitive care provin de la fabricanți diferiți. Modulul `sw_script` a fost folosit în cadrul tester-ului GETB pentru a verifica integritatea rezultatelor (măsurătorile efectuate de tester erau comparate cu statisticile înregistrate de dispozitivului testat).

Tot în cadrul fazei de instalare, autorul a contribuit cu un sistem pentru inventarierea echipamentelor și conexiunilor din rețea. A fost conceput și implementat un algoritm care descoperă topologia rețelei (Secțiunea 5.2). Astfel ne-am putut asigura că instalarea cablurilor și a echipamentelor a fost efectuată conform planurilor.

Darea în funcțiune – Monitorizarea traficului Din 2008, dată la care va începe experimentul ATLAS, rețeaua va fi monitorizată în permanență – orice defecțiune va trebui rezolvată într-un timp cât mai scurt. Folosind harta rețelei (topologia) autorul a realizat un sistem pentru vizualizarea în timp real a gradului de ocupare al legăturilor dintre switch-uri. Astfel, utilizatorul (administratorul de rețea) poate observa foarte ușor cum este folosită rețeaua și care sunt punctele cu un nivel de încărcare ridicat.

Atunci când apar probleme cauzate de trafic, primul pas spre rezolvare constă în determinarea aplicațiilor sau a stațiilor care produc congestia. Tehnologia *sFlow*, bazată pe eșantionarea pachetelor, poate fi de un real folos în astfel de situații. Autorul a realizat un studiu privitor la *sFlow* (Capitolul 6). A fost realizată o aplicație de analiză *sFlow* și au fost prezentate tipurile de rezultate ce pot fi obținute folosind eșantionarea. De asemenea, au fost analizate cazuri particulare care au relevat limitele acestei tehnologii.

7.2 Perspective

Pentru viitor se conturează două direcții de cercetare:

Vizualizarea Afișarea informației pentru o rețea de mari dimensiuni reprezintă o temă activă de cercetare. La ora actuală sunt investigate două posibilități: una se bazează pe afișarea rețelei ca un graf bidimensional iar a doua pe construirea unui mediu interactiv 3D pentru întregul sistem de achiziție TDAQ. Prima abordare scoate în evidență relațiile între elementele de rețea, în timp ce a doua este mai potrivită pentru reprezentarea legăturilor logice dintre componentele sistemului de achiziție și filtrare.

Sisteme de diagnosticare Când sistemul de achiziție nu funcționează conform așteptărilor, este foarte important ca problemele să fie rezolvate rapid și eficient. Se preconizează implementarea unui “sistem expert” care să coreleze informații provenite din diferite surse (cea mai importantă fiind sistemul de gestiune a rețelei). Atunci când apare o eroare, sistemul ar trebui să prezinte operatorului o listă cu cauzele cele mai probabile și eventual cu posibile soluții.

Bibliografie selectivă

- [1] ATLAS HLT/DAQ/DCS Group, “ATLAS High-Level Trigger Data Acquisition and Controls Technical Design Report, *CERN/LHCC/2003-022*, October 2003. [Online]. Available: <http://atlas-proj-hltDAQDCS-tdr.web.cern.ch/atlas-proj-hltDAQDCS-tdr/>
- [2] R. Dobinson, M. Dobson, S. Haas, B. Martin, M. J. LeVine, and F. Saka, “IEEE 802.3 Ethernet, Current Status and Future Prospects at the LHC, *CERN open-2000-311*, October 2000. [Online]. Available: <http://doc.cern.ch/archive/electronic/cern/preprints/open/open-2000-311.pdf>
- [3] S. Stancu, M. Ciobotaru, B. Dobinson, K. Korcyl, and E. Knezo, “The use of Ethernet in the Dataflow of the ATLAS Trigger and DAQ, in *Computing in High Energy and Nuclear Physics*, La Jolla, California, 2003. [Online]. Available: http://cern.ch/ciobota/papers/2003_ethernet_for_atlas.pdf
- [4] S. Stancu, M. Ciobotaru, and K. Korcyl, “ATLAS TDAQ DataFlow Network Architecture Analysis and Upgrade Proposal, in *IEEE Trans. on Nuclear Science*. IEEE Nuclear and Plasma Sciences Society, 2006, vol. 53, No. 3, pp. 826–833. [Online]. Available: http://cern.ch/ciobota/papers/2005_atlas_net_proposal.pdf
- [5] S. Stancu, M. Ciobotaru, C. Meirosu, L. Leahu, and B. Martin, “Networks for the ATLAS Trigger and Data Acquisition, in *Computing in High Energy and Nuclear Physics*, Mumbai, India, 2006. [Online]. Available: http://cern.ch/ciobota/papers/2006_networks_tdaq.pdf
- [6] S. Stancu, M. Ciobotaru, and D. Francis, “Relevant features for DataFlow switches, April 2005. [Online]. Available: http://cern.ch/ciobota/papers/2005_switch_features.pdf
- [7] M. Ciobotaru, S. Stancu, M. J. LeVine, and B. Martin, “GETB, a Gigabit Ethernet Application Platform: its Use in the ATLAS TDAQ Network, in *IEEE Trans. on Nuclear Science*. IEEE Nuclear and Plasma Sciences Society, 2006, vol. 53, No. 3, pp. 817–825. [Online]. Available: http://cern.ch/ciobota/papers/2005_getb_stockholm.pdf
- [8] S. M. Batraneanu, A. Al-Shabibi, M. Ciobotaru, M. Ivanovici, L. Leahu, B. Martin, and S. Stancu, “Operational Model of the ATLAS TDAQ Network, in *Proc. IEEE Real Time 2007 Conference*, Chicago, USA, May 2007. [Online]. Available: http://cern.ch/ciobota/papers/2007_operational_model.pdf
- [9] Wikipedia, “Netconf – The Network Configuration Protocol. [Online]. Available: <http://en.wikipedia.org/w/index.php?title=Netconf&oldid=123491664>
- [10] Y. Bejerano, “A Simple And Efficient Topology Discovery Scheme For Large Ethernet LANs, in *Proc. of IEEE INFOCOM*, 2006. [Online]. Available: http://www1.bell-labs.com/user/bejerano/Papers/sklt_tree.L2_net_disc_IC06V2.pdf
- [11] E. Adar, “GUESS: A Language and Interface for Graph Exploration. [Online]. Available: <http://graphexploration.cond.org/chi2006/guess-chi2006.pdf>
- [12] “RFC 3176 - InMon Corporation sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. [Online]. Available: <http://www.faqs.org/rfcs/rfc3176.html>