Universitatea “Politehnica” Bucureşti

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Structura unui switch

Conducător ştiinţific:

Conf. Dr. Ing. Ştefan Stăncescu

Masterand: Ing. Ignat D. Mihai

2015

Cuprins

Introducere

Rețelele de calculatoare au cunoscut o dezvoltare rapidă în ultima jumătate de secol. Pornind de la rețelele bazate pe comutare de circuite, care utilizau infrastructura tradițională de telefonie pentru trimiterea datelor și se bazau pe stabilirea unei conexiuni electronice punct-la-punct între stațiile comunicante, rețelele au cunoscut un salt considerabil odată cu apariția comutării de pachete. Aceasta presupune transmiterea datelor în pachete de dimensiune cunoscută în rețea, acestea ajungând la destinație pe căi partajate de mai multe sesiuni de comunicare. Printre avantajele acestei metode de transmisie se numără creșterea eficienței rețelei, a robusteței acesteia și posibilitatea de a transmite simultan către mai multe stații. ARPAnet este printre primele rețele care au utilizat această metodă, aceasta implementând și protocolul TCP/IP pentru prima dată.

Protocolul TCP/IP a fost dezvoltat pornind de la modelul OSI (Open Systems Interconnection). Acesta prevede 7 nivele logice de comunicare, iar la nivelul fiecărui nivel au fost dezvoltate o serie de protocoale.

Fiecare echipament de rețea reprezintă o implementare corespunzătoare a stivei OSI sau mai concret, a modelului real de protocol de comunicație implementat. Spre exemplu, o stație de lucru conectată la o rețea care implementează protocolul TCP/IP va avea prevăzută implementarea tuturor nivelelor acestui model, în timp ce un echipament de rețea precum un router va avea prevăzută implementarea primelor două nivele ale protocolului.

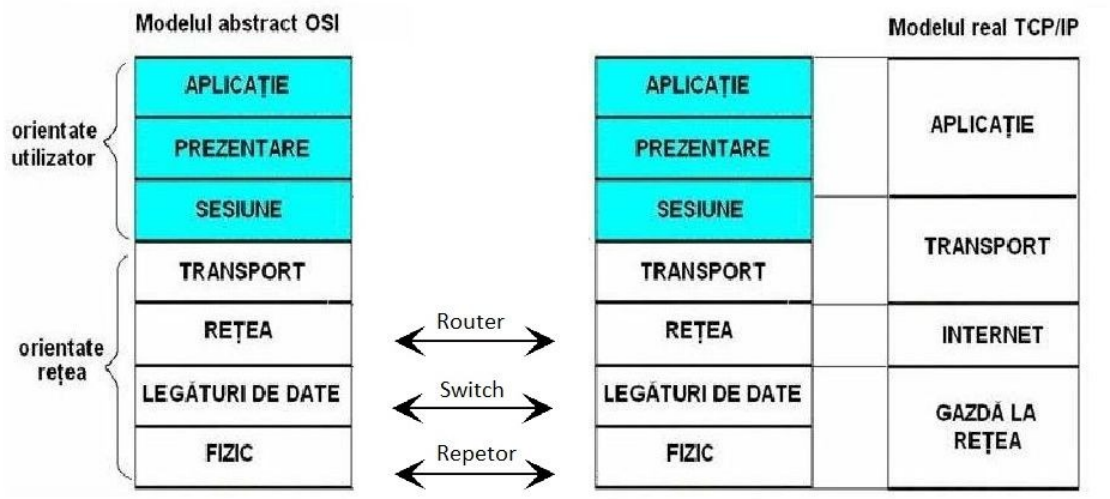

Figura 1 – Maparea dintre modelul OSI si modelul TCP/IP [1]

În figura de mai sus, pe lângă corespondențele între modelul OSI și cel TCP/IP, a fost evidențiată și corespondența între echipamentele de rețea care operează la nivelele respective.

Nivelul fizic prevede specificațiile electrice și fizice ale conexiunii de date și relația dintre un dispozitiv și mediul de transmisie fizic (cablu de cupru sau fibră optică). Un

dispozitiv care operează la acest nivel nu necesită capacitatea de a decoda pachetele primite prin semnalele respective. Astfel de dispozitive se numesc repetoare sau hub-uri. Funcția de bază a acestora este de a replica un semnal recepționat. [2]

La nivel de legătură de date sunt prevăzute transferurile de la nod la nod, cu scopul de a obține o legătură de încredere între două noduri conectate direct, prin detecția și corecția erorilor apărute în nivelul fizic. Acest nivel este divizat în subnivelul MAC (Media Access Control), care controlează cum obțin dispozitivele din rețea accesul la date și LLC (Logical Link Control), care controlează verificarea erorilor și sincronizarea pachetelor. [2]

Proiectarea unui switch

Unul dintre echipamentele care operează la nivelul legăturii de date este switch-ul de rețea (figura 1).

Figura 2 – Switch-ul Avaya ERS 2550T-PWR [3]

Un switch de rețea este un dispozitiv care conectează împreună într-o rețea de calculatoare mai multe echipamente, utilizând comutarea de pachete pentru a primi, procesa și transmite mai departe datele către dispozitivele destinatare. Diferența majoră între un dispozitiv de nivel fizic și un switch constă în faptul că un switch trimite mai departe datele către unul sau mai multe dispozitive destinatare în loc de a transmite aceleași date pe toate porturile. [3]

Ethernet este un mediu partajat de mai mulți utilizatori, așadar este necesar un set de reguli pentru transmiterea pachetelor evitând coliziunile și protejând integritatea datelor. Nodurile dintr-o rețea Ethernet trimit pachetele atunci când determină faptul că rețeaua nu este utilizată. Este posibil ca două noduri diferite din rețea să încerce să transmită date în același timp, rezultând o coliziune. Ambele pachete vor fi retransmise, crescând astfel traficul din rețea. Minimizarea coliziunilor este un element crucial în proiectarea și operarea rețelelor. Un număr mare de coliziuni este rezultatul a unui număr prea mare de utilizatori sau a unei cantități prea mare de trafic în rețea, care duce la creșterea benzii de rețea utilizată. Acest lucru poate duce la scăderea performanței rețelei din punctul de vedere al utilizatorului. Utilizarea segmentelor de rețea, îmbinate logic prin switch-uri sau router-e duce la reducerea congestiei într-o rețea supraaglomerată. [4]

Switch-urile mapează adresele Ethernet ale nodurilor corespunzătoare de pe un segment de rețea și apoi permit trecerea prin switch doar a traficului necesar. Când switch-ul primește un pachet, acesta examinează adresele hardware ale sursei și ale destinației și le compară cu un tabel de adrese și segmente de rețea. Dacă segmentul de rețea al destinației este același cu cel al sursei, pachetul recepționat este ignorat (”filtered”); dacă segmentele sunt diferite, atunci pachetul este transmis mai departe (”forwarded”) către segmentul corespunzător. Adițional, switch-urile previn răspândirea pachetelor malițioase sau nealiniate prin blocarea lor. [4]

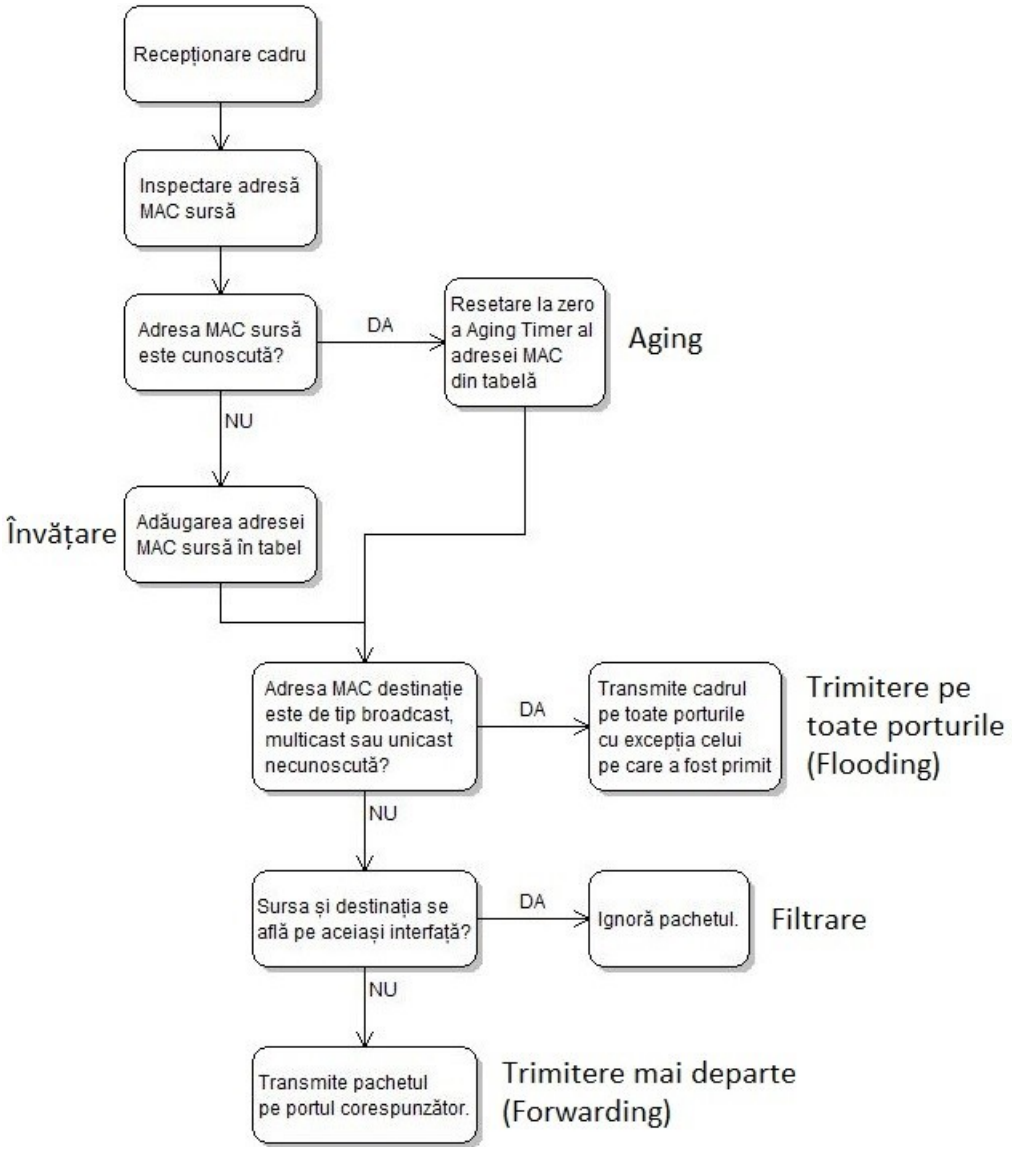

Figura 3 – Schema bloc de procesare a unui pachet utilizând un switch [5]

În figura de mai sus pot fi observate mai multe stagii de procesare a unui pachet [5]:

1. Învățarea (learning) – switch-ul examinează adresa MAC sursă din cadrul recepționat și o asociază cu portul pe care a fost primită (ingress) prin adăugarea unei intrări în tabelul de adrese MAC;

2. Trimiterea pe toate porturile (flooding) – switch-ul examinează adresa MAC destinație, apoi o compară cu intrările din tabela de adrese MAC și în cazul în care aceasta nu este găsită, cadrul respectiv este trimis pe toate porturile (mai puțin cel de ingress) pentru a se asigura recepționarea cadrului de către destinatar. Există o clasă speciale de adrese MAC de broadcast, care sunt concepute astfel încât switch-urile care primesc pachete conținând astfel de adrese vor trimite mai departe pe toate porturile sau pe o selecție de porturi de egress, dacă switch-ul suportă controlul traficului de multicast;

3. Trimiterea mai departe a cadrului (forwarding) – switch-ul examinează adresa MAC sursă a pachetului primit și o înregistrează în tabelul de adrese MAC, asociind adresa de portul pe care a fost primită. Acesta apoi examinează adresa MAC destinație, căutând în tabelă și găsește adresa respectivă asociată unui port. Astfel, switch-ul trimite mai departe cadrul pe portul respectiv de egress.

4. Filtrarea (filtering) – la examinarea unui cadru, dacă switch-ul determină faptul că într-un cadru, adresele MAC sursă și destinație aparțin aceluiași port, atunci acesta ignoră pachetul și nu-l trimite mai departe.

5. ”Îmbătrânirea” (aging) – la introducerea unei intrări în tabela de adrese MAC, acesteia i se asociază un contor pentru a monitoriza inactivitatea adresei respective. La fiecare cadru recepționat cu adresa respectivă, contorul este resetat. În cazul în care nu se mai realizează trafic pe rețea cu adresa MAC respectivă, la depășirea unei valori de timp limită, intrarea pentru acea adresă MAC în tabelă este ștearsă pentru a elibera resursele de memorie și pentru a asigura schimbările de topologie.

Printre beneficiile utilizării unui switch se regăsesc [4]:

- izolarea traficului și reducerea congestiei
- separarea domeniilor de coliziune și implicit reducerea coliziunilor
- segmentarea rețelei

Dezavantajele utilizării unui switch constă în [4]:

- preț ridicat, prin comparație cu un hub
- procesarea pachetelor aduce o latență suplimentară în rețea
- monitorizarea rețelei crește în complexitate

Abordarea hardware

Printre primele switch-uri apărute în comerț se regăsesc modele precum Kalpana EtherSwitch EPS-1500 (figura 4).

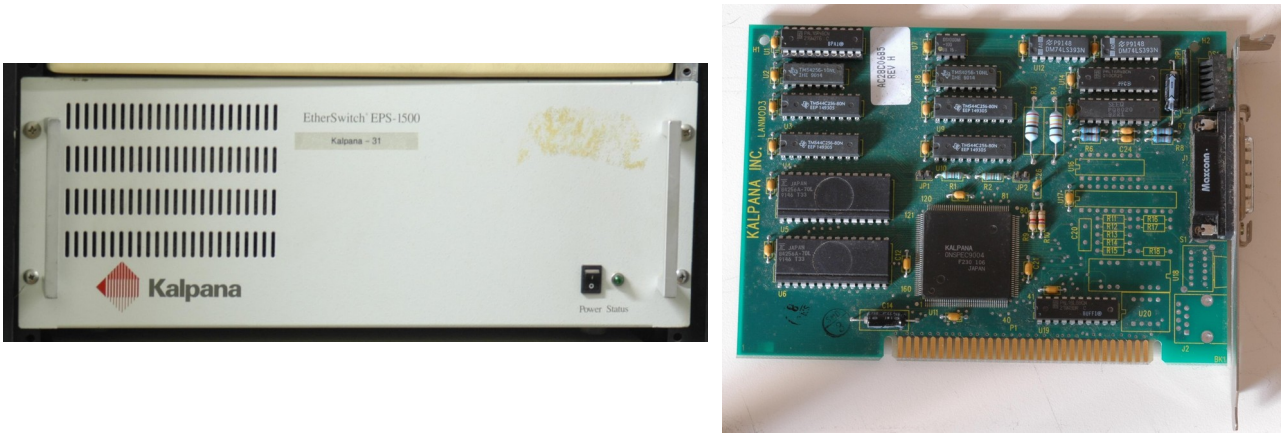

Figura 4 – Switch-ul Kalpana EtherSwitch EPS-1500 (stânga)

și placa de bază a acestuia (dreapta) [7]

Acesta este printre primele switch-uri Ethernet apărute pe piață. Din poza plăcii de bază se pot observa module de memorie RAM statice (84256A-70L) și dinamice (TMS44C256), circuite integrate cu rol de contor (DMZ4L5393N), circuite integrate de tip PLA (PAL16R4BCN), circuite de tip convertoare de semnal (SEEQ PQ8020) și microprocesoare de aplicație (KALPANA ONSPEC9004).

Fiecare componentă are un rol specific în cadrul sistemului. Circuitele de tip convertor de semnal realizează decodarea semnalului primit pe porturi, circuitul PLA se ocupă de procesarea pachetelor, circuitele de tip contor sunt utilizate pentru funcția specială de aging, memoriile sunt utilizate pentru stocarea tabelei de adrese MAC iar procesorul de aplicație se ocupă de managementul celorlalte componente.

Pe baza acestor detalii, se poate detalia schema bloc din figura următoare.

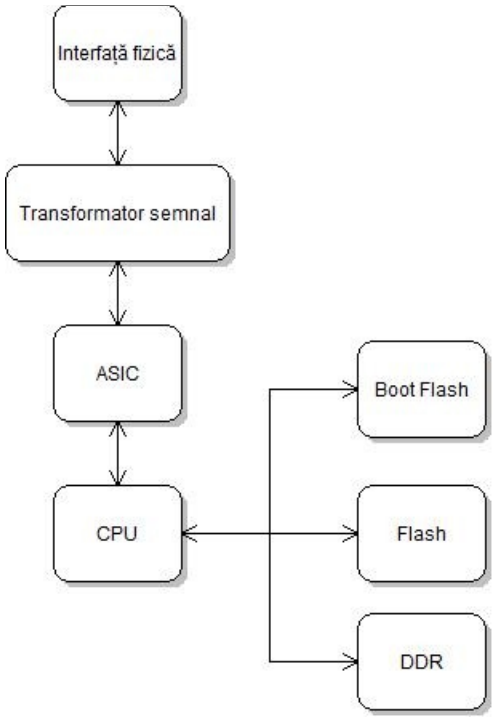

Figura 5 – Schema hardware bloc a unui switch

Interfața fizică a switch-ului reprezintă legătura cu rețeaua în care este legat. Aceasta este specifică tehnologiei de rețea utilizate. Cea mai populară este Ethernet, cu interfața RJ45, observabilă în figura 1. Alte interfețe utilizate sunt GBIC și InfiniBand (figura 6), pentru conectarea la fibra optică.

Figura 6 – Interfața GBIC (stânga) [8]

și interfața InfiniBand [9]

În timp ce GBIC este utilizat în mod curent în rețelele de calculatoare, InfiniBand este preponderent utilizată în rețelele de supercomputere.

Transformatorul de semnal se ocupă de transformarea semnalului primit pe interfața fizică în semnal util pentru logica circuitelor ce urmează.

Circuitul ASIC (Application Specific Integrated Circuit) are aceiași funcție precum circuitul PLA din switch-urile inițiale. Acesta se ocupă cu procesarea pachetelor și menținerea integrității tabelei de adrese MAC, printre altele.

Unitatea centrală de procesare este cea care se ocupă de managementul general al switch-ului, precum și de rularea diferitelor aplicații specifice acestuia. Acesta necesită memorie pentru pornire (boot), memorie flash pentru stocarea codului aplicațiilor și a diferitelor loguri de sistem, printre altele, dar și de memorie DDR necesară atât la rularea funcțiilor de sistem, cât și la rularea aplicațiilor.

Abordarea software

Precum și alte sisteme de tip embedded, switch-urile au evoluat de la sisteme simple, lipsite de management-ul unui sistem de operare sau având un sistem de operare care se ocupă de operațiile de bază, la sisteme din ce în ce mai complexe, conforme cu cerințele industriei. Numărul de aplicații pe care switch-ul le poate rula a crescut considerabil, ducând la ineficiență un sistem de management simplu al resurselor. Necesitatea unui sistem de operare complex pentru platforma switch-ului depinde și de utilizarea acestuia. Un switch utilizat de către un simplu utilizator nu va rula un număr mare de aplicații complexe, iar acesta nu va necesita un sistem de operare complex. În schimb, un switch utilizat într-un mediu de tip corporate, precum Avaya ERS 2550T-PWR necesită un management precis, deoarece orice pierdere de date poate duce la probleme serioase. Printre cele mai populare sisteme de operare pentru astfel de echipamente embedded se regăsește și VxWorks.

Majoritatea switch-urilor rulează servicii specifice nivelului de legătură de rețea precum încapsularea pachetelor de date primite de la nivelul de rețea în cadre, sincronizarea cadrelor, controlul erorilor (ARQ, FEC), controlul direcției de transmisie (flow control, utilizat preponderent în modemuri și rețele wireless), protocoale de acces multiplu la mediu (CSMA/CD pentru detecția coliziunilor și retransmisia în rețea sau CSMA/CA pentru evitarea coliziunilor în rețelele wireless), adresarea fizică, comutarea de pachete (incluzând filtrarea MAC, STP și SPB), comutarea de tip stocare și transmitere mai departe, controlul QoS și VLAN. [6]

Switch-urile concepute pentru mediile corporate permit în plus rularea unor aplicații de nivel superior precum OSPF , IGMP snooping, IP multicast (nivelul de rețea), servicii de firewall, IPSec, VPN (nivelul de transport) și în unele cazuri, există posibilitatea

implementării unei memorii cache pentru stocarea celor mai accesate site-uri (nivelul de aplicație). [3]

ASIC-urile sunt cele care realizează procesarea efectivă a cadrelor. Acest lucru este realizat la nivel hardware, ASIC-ul fiind preconfigurat pentru a executa procesarea. Acesta menține datele la nivelul hardware-ului său. Aplicațiile necesită managementul OS-ului și implicit driverele implementate pentru a accesa datele hardware ale ASIC-ului.

Schema bloc a dependențelor dintre hardware și software într-un switch este ilustrată în figura următoare:

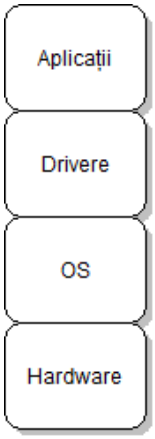

Figura 7 – Schema bloc a dependențelor HW-SW în switch

Driverele reprezintă modalitatea de acces în hardware. Acestea mapează funcțiile corespunzătoare ale sistemului de operare pentru a accesa memoria ASIC-ului și pentru a controla comportamentul acestuia.

Aplicațiile își gestionează propriul spațiu de memorie și utilizează driverele pentru a-și menține consistența datelor în raport cu hardware-ul ASIC. Pe baza acestor date, acestea trimit comenzi corespunzătoare tot prin intermediul driverelor, către hardware-ul de execuție (ASIC).

VxWorks

Sistemele de operare utilizate pentru switch-uri sunt sistemele de operare în timp real. Necesitatea acestora în astfel de aplicații se datorează faptului că procesarea cadrelor recepționate de către echipament trebuie să fie imediată, fără întârziere, timpul necesar prelucrării pachetelor fiind de ordinul zecimilor de secundă sau mai puțin.

Diferențele dintre un sistem de operare de uz general și un sistem de operare în timp real sunt următoarele [10]:

1. Determinismul operațiilor – comportarea în timp a unui sistem în timp real este cunoscută apriori, sistemul de operare lucrând în perioade de timp cunoscute. Latența unui sistem de operare în timp real este cunoscută. Prin comparație, un sistem de operare de uz general nu ține cont de latență;

2. Planificarea task-urilor – un sistem de operare de uz general asigură timp de procesare pentru fiecare task din sistem, astfel asigurându-se că și task-urile de prioritate scăzută primesc timp de execuție, planificarea fiind bazată pe proces. Un sistem de operare în timp real utilizează o planificare preemptivă bazată pe prioritate care permite task-urilor cu prioritate mare să fie executate la timp, în mod determinist. Acest lucru este important într-un sistem embedded deoarece o întârziere poate cauza defecțiuni majore;

3. Kernel preemptiv – sistemele de operare în timp real utilizează operații de kernel preemptive;

4. Inversarea priorităților – sistemele de operare în timp real au mecanisme de prevenire a inversării priorităților;

5. Utilizare – sistemele de operare în timp real sunt utilizate în general în aplicații embedded, în timp ce sistemele de operare de uz general sunt proiectate pentru sisteme desktop și alte sisteme similare.

Un sistem de operare în timp real trebuie să garanteze prelucrarea în acest timp prestabilit. Conform acestui criteriu, există următoarea clasificare a sistemelor de operare în timp real [10][11]:

- sisteme în timp real stricte (hard) – o ratare de execuție a unui proces duce la căderea totală a sistemului;

- sisteme în timp real ferme (firm) – ratările de execuție sunt tolerate dacă sunt nefrecvente, dar în timp acestea vor duce la degradarea calității serviciilor sistemului, iar utilizatea rezultatului ratat este zero;

- sisteme în timp real fără constrângeri (soft) – ratările duc la rezultate inutile care duc la degradarea calității serviciilor sistemului.

Un sistem de operare în timp real care se bucură de succes comercial este VxWorks, dezvoltat de compania Wind River. Apărut în 1987, acesta a fost dezvoltat pentru utilizarea în sisteme embedded care necesită performanțe deterministice în timp real și certificări de siguranță și securitate, precum dispozitive medicale, echipamente industriale, roboți, diverse utilizări în industriile aerospațiale și de apărare, ș.a.m.d. Pachetul de dezvoltare VxWorks include kernel-ul, middleware-ul, pachetele de suport specifice plăcilor, pachetul de dezvoltare Wind River Workbench și tehnologii software și hardware complementare de la alți dezvoltatori. Ultima versiune de VxWorks este versiunea 7, care prevede o modularitate și o trecere la o versiune superioară regândită astfel încât kerne-ul de sistem este separat de restul pachetului. De asemenea, există îmbunătățiri legate de scalabilitate, securitate, siguranță, conectivitate și grafică conforme cu ultimele dezvoltări ale Internetului Lucrurilor (Internet of Things – IoT). [12]

VxWorks oferă, printre altele [12]:

- kernel multitasking cu planificare preemptivă și round-robin și răspuns rapid la întreruperi;
- suport pentru arhitecturi native de 64 de biți;
- aplicații în mod utilizator izolate de alte aplicații de utilizator și de kernel prin mecanisme de protecție a memoriei;
- suport pentru moduri de multiprocesare SMP, AMP și mixte;
- framework pentru managementul erorilor;
- suport pentru protocoalele Bluetooth, USB, CAN, Firewire, BLE, s.a.m.d.;
- semafoare binare, de tip numărător și MuTex cu moștenirea priorităților;
- cozi de mesaje distribuite și locale;
- certificare POSIX PSE52;
- sisteme de fișiere HRFS, FAT, NFS, TFFS;
- certificare IPv6;
- protecția memoriei incluzând procesele în timp real, detecția și raportarea erorii și IPC (apelurile interproces);
- transmiterea între mai multe sisteme de operare utilizând TIPC și IPC-uri Wind River multi-OS;
- debugging utilizând simboluri.

Dintre utilizările notabile ale sistemului VxWorks, reamintim [12]:

- vehiculele de teren pentru explorarea planetei Marte (Sojourner, Spirit, Opportunity);
- satelitul de explorare spațială Deep Impact;
- racheta SpaceX Dragon;
- sistemele de bază ale avionului Boeing 787;
- sistemul aerian fără pilot Northrop Grumman X-47B;
- elicopterul de atac Boeing AH-64 Apache;
- telescopul spațial Fermi Gamma-ray;
- sistemul de navigare auto Siemens VDO;
- sistemul de telemetrie auto Bosch Motor Sports;
- robotul Honda ASIMO;
- gama de switch-uri Avaya ERS și VSP;

Arhitectura kernel-ului VxWorks

Versiunile anterioare ale sistemului de operare VxWorks ofereau un singur spațiu de memorie fără diferențiere între sistemul de operare și aplicațiile utilizatorului. Toate task-urile rulau în mod supervizor. Deși acest model era potrivit pentru dezvoltarea aplicațiilor, numai un programator foarte bun putea asigura coexistența între facilitățile kernelului și aplicații fără interferențe în același spațiu de memorie. Odată cu versiunea 6, execuția aplicațiilor a trecut în modul utilizator, delimitând clar kernel-ul de aplicații. Acest model de arhitectură este referențiat des ca fiind modeulul de proces. VxWorks a adoptat acest model având în vedere cpnstrângerile sistemelor de timp real, oferind protecție bazată pe MMU pentru spațiile de utilizator și de kernel. De asemenea, versiunea 6 oferă compatibilitate cu versiunile anterioare, existând posibilitatea de migrare a aplicațiilor dezvoltate către această versiune cu efort minim. Dezvoltarea de aplicații noi se poate realiza pentru spațiul kernel-ului luând în considerare următoarele [13]:

- spațiul ocupat – spațiul ocupat de sistem este mai mic fără componentele care oferă suport pentru procese și MMU;
- viteză – rularea în kernel poate fi mai rapidă, depinzând de numărul de apeluri sistem pe care aplicația le face și de câte operații I/O sunt efectuate ca proces în spațiul utilizatorului;

- facilități exclusive de kernel – facilități precum watchdog timers, ISR-uri și VxMP sunt disponibile numai șla nivelul kernel-ului, dar există și alternative pentru aplicații bazate pe proces (POSIX timers);
- accesul la hardware – dacă aplicația necesită acces direct în hardware, acesta poate fi realizat doar din kernel.

VxWorks oferă flexibilitate în modularitatea facilităților și extensiilor sale. Astfel, acesta poate fi configurat să includă doar un planificator de task-uri, managementul întreruperilor, managementul memoriei dinamice și alte facilități minime, sau poate fi configurat având componente pentru execuția aplicațiilor ca procese, gestiunea sistemelor de fișiere, interfețe de rețea, detecția și raportarea erorilor, ș.a.m.d. Și alte elemente particularizate de kernel pot fi incluse, ca de exemplu, componente pentru noi sisteme de fișiere sau protocoale de rețea, etc. Interfața de apel de sistem poate fi extinsă prin adăugarea de API-uri create special pentru aplicații bazate pe proces [13].

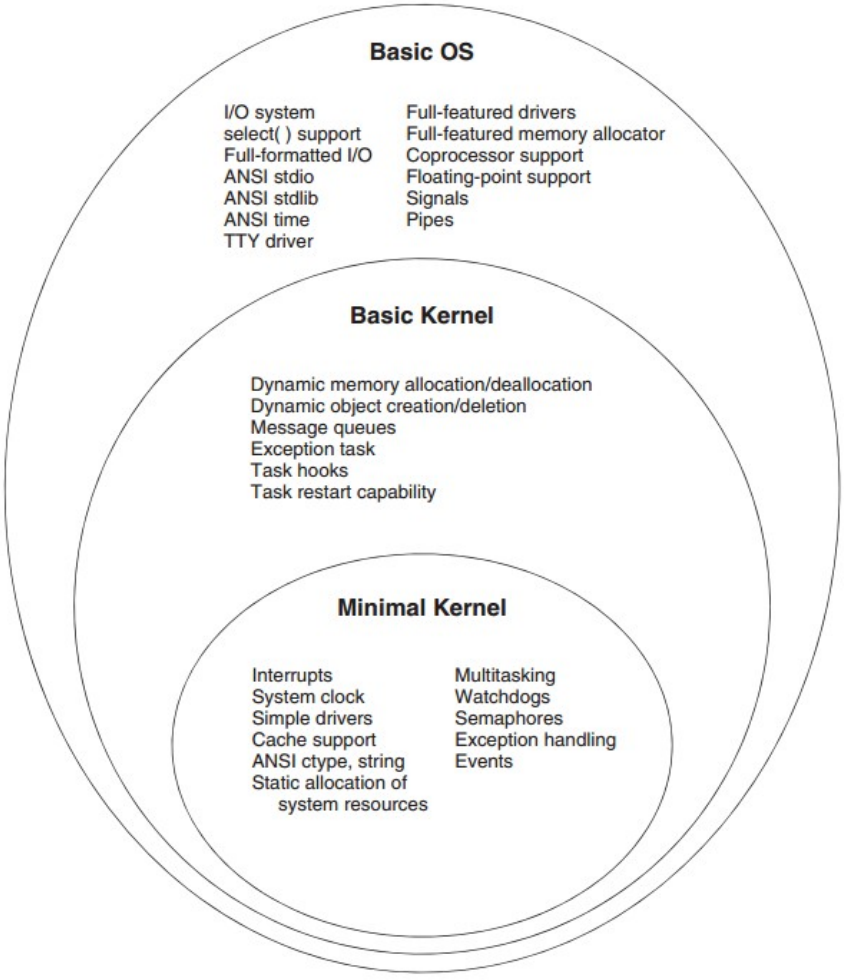

Figura 8 – Structura kernel-ului VxWorks [13]

Figura 8 evidențiază modularitatea kernel-ului VxWorks. Fiecare nivel oferă facilități în funcție de necesitățile aplicației.

Modelul minim oferă cel mai mic nivel de operare pentru VxWorks, constând în suportul de bază pentru CPU și BSP, semafoare și watchdog timers. Kernel-ul este un sistem static, neexistând suport pentru alocarea dinamică a memoriei. Atât aplicațiile cât și sistemul trebuie sa aibă componentele declarate la compilare. Lipsa de facilități de memorie dinamică implică faptul că kernel-ul nu poate instanția dinamic obiecte precum task-uri, semafoare și timere watchdog. De asemenea, la nivel de drivere, nu există suport pentru funcții tradiționale de acces I/O precum open(), close(), read(), write(), etc. [13]

Modelul kernel de bază include facilitățile oferite de modelul minim, aducând în plus comunicarea inter-task prin cozi de mesaje, suport pentru task hooks, alocarea și eliberarea memoriei, abilitatea de creare și ștergere dinamică a obiectelor kernel precum task-uri, semafoare, timere watchdog și cozi de mesaje, suport pentru rutine ANSI precum strdup(). [13]

Modelul Basic OS oferă în plus facilități precum sistemul I/O, suportul pentru descriptorii de fișier I/O standard și API-ul asociat, API-uri pentru manipularea directoarelor și a căilor și utilitare de disc, suport pentru select(), suport pentru TTY și drivere Pipe, suport pentru logging, suport pentru variabile de task și mediu, suport pentru management-ul coprocesorului, manager de partiționarea memoriei cu facilități complete, suport complet pentru librăria ANSI. [13]

Gestionarea unui switch prin SNMP

Simple Network Management Protocol (SNMP) reprezintă un protocol de aplicație definit de către Internet Architecture Board (IAB) în RFC1157 pentru schimbul de informații de management între dispozitive de rețea și face parte din suita de protocoale TCP/IP. Acest protocol este unul dintre cele mai răspândite protocoale pentru managementul și monitorizarea elementelor de rețea. Majoritatea echipamentelor profesioniste includ facilități SNMP. Pentru a putea comunica cu sistemul de management al rețelei (NMS), acestea trebuie să fie pornite și configurate. În prezent, versiunea existentă este SNMPv3, în care sunt definite mecanisme de securitate. [14]

Protocolul SNMP este compus din [14][15]:

- managerul SNMP;
- dispozitivele controlate;
- agentul SNMP;
- baza de informație de management (Management Information Base – MIB).

Managerul SNMP este o entitate separată responsabilă cu comunicarea cu agentul SNMP implementat în dispozitivul de rețea. Acesta este în mod uzual un computer utilizat pentru a rula unul sau mai multe sisteme de management al rețelei. Funcțiile cheie ale managerului sunt [14]:

- interogarea agenților;
- recepționarea răspunsului de la agenți;
- setarea variabilelor în agenți;
- recepționarea evenimentelor asincrone de la agenți.

Dispozitivele controlate sau elementele de rețea sunt părți de rețea care necesită o formă de monitorizare și management, ca de exemplu routere, switch-uri, servere, stații de lucru, imprimante de rețea, UPS-uri, etc.

Agentul SNMP este un program care este incorporat în elementul de rețea. Pornirea agentului îi permite colectarea bazei de informații de management de la dispozitivul local și pune la dispoziție datele către managerul SNMP care a realizat interogarea. Acești agenți poți fi standard (Net-SNMP) sau specific vendorilor (SR Research, Envoy). Funcțiile cheie ale agentului SNMP sunt [14]:

- colectarea informațiilor de management legate de mediul local în care rulează;
- stocarea și preluarea informațiilor de management precum sunt definite în MIB;

- semnalarea evenimentelor către manager;
- acționarea ca proxy pentru noduri de rețea manevrabile prin mecanisme non-SNMP.

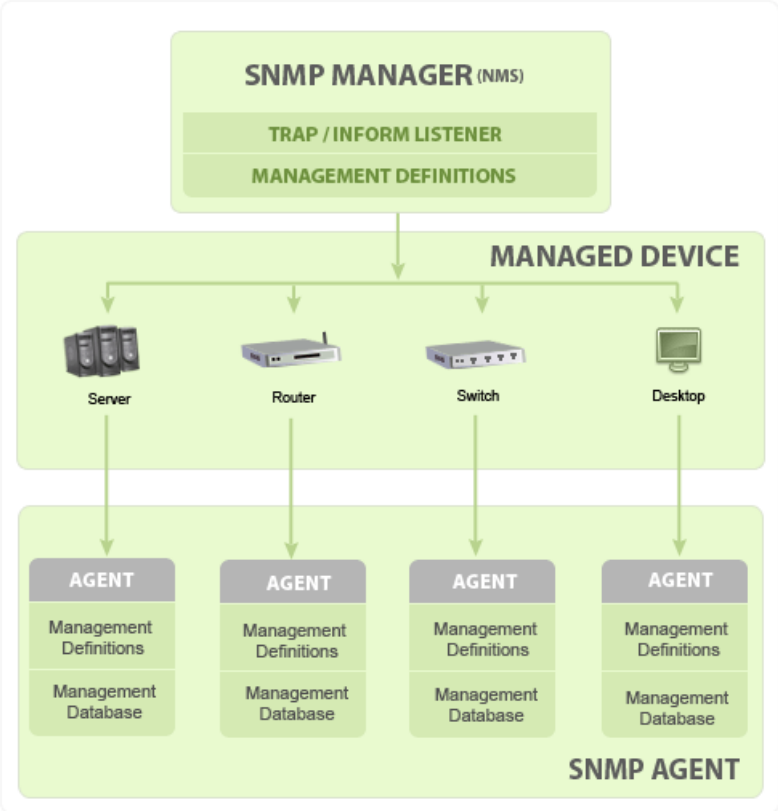

Figura 9 – Diagrama de comunicație de bază SNMP [14]

Fiecare agent SNMP menține o bază de date de informații care descriu parametrii dispozitivului controlat. Managerul SNMP utilizează această bază de date pentru a trimite cereri către agent pentru informații specifice și translatează mai departe informația către NMS precum este cerută. Această bază de date partajată între agent și manager se numește baza de informații de management (MIB). În mod uzual, MIB-urile conțin un set standard de valori statistice și valori de control definite pentru nodurile hardware din rețea. SNMP permite extinderea acestor valori standard cu valori specifice unui agent particular prin utilizarea MIB-urile private. Ca o paralelă, fișierele MIB sunt un set de întrebări pe care managerul SNMP le pune agentului. Agentul colectează datele local și le stochează, precum sunt definite în MIB. Astfel, managerul SNMP ar trebui să poată pune aceste întrebări pentru fiecare agent. [14][15]

MIB-ul este o colecție de informații utilizate la managementul elementului de rețea. MIB-urile sunt formate din obiecte controlabile identificabile prin identificatorul de obiect (OID). Fiecare identificator este unic și denotă carecteristicile specifice ale dispozitivului controlat. La interogare, valoarea returnată pentru fiecare identificator poate fi diferită

(număr, contor, text, etc.) Există două tipuri de obiecte controlabile sau OID-uri: scalare și tabulare. Spre exemplu, un obiect scalar este numele vendorului dispozitivului, care este o singură instanță. Un obiect tabular poate fi utilizarea procesorului pentru un procesor cu patru nuclee. Rezultatul unei astfel de interogări va returna 4 valori separate, pentru fiecare nucleu în parte, sub același OID. Fiecare OID este organizat ierarhic în MIB, structura acestuia putând fi reprezentată sub forma unui arbore, fiecare nod reprezentând un identificator de variabilă individual. Spre exemplu, pentru OID-ul scalar ”sysDescr”, calea în arbore va fi o listă punctată de întregi de tipul .1.3.6.1.2.1.1.1, reprezentată grafic în figura următoare. [14]

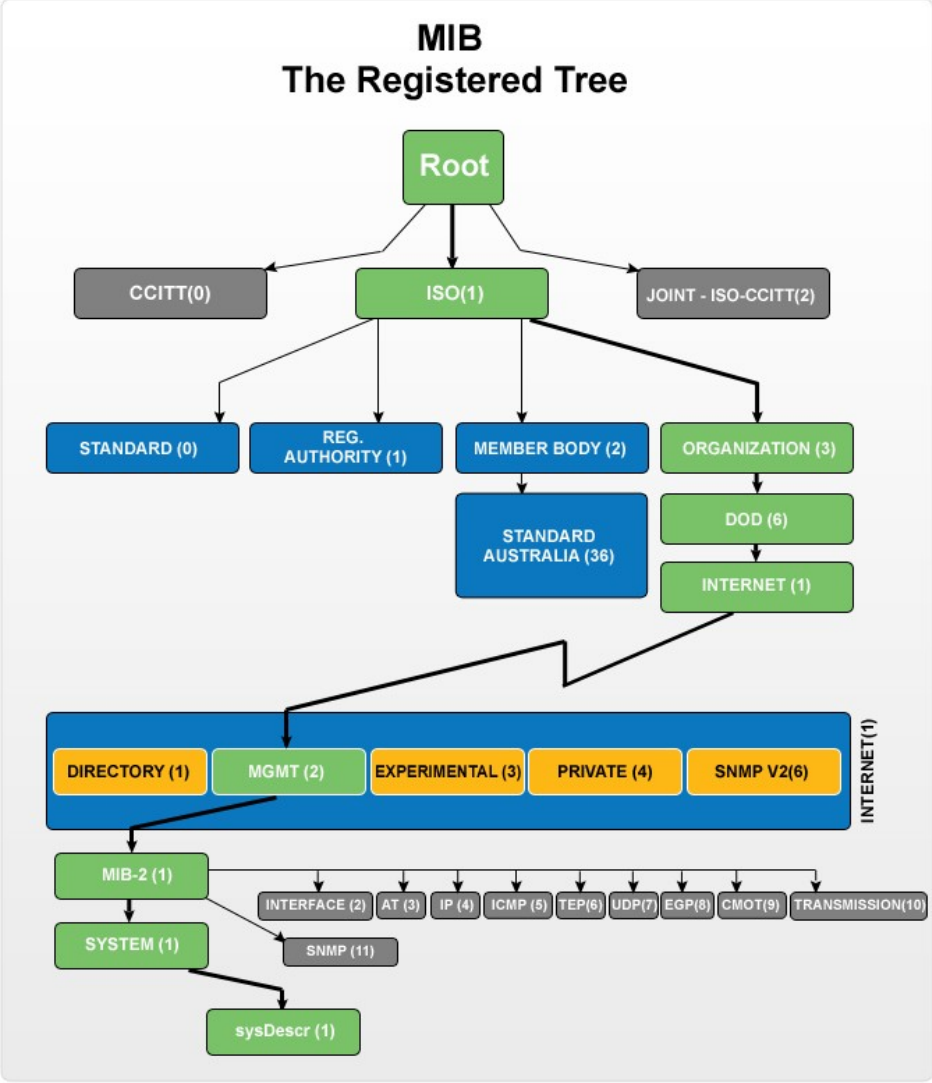

Figura 10 – Reprezentarea grafică a MIB-ului pentru OID-ul sysDescr [14]

Comenzile de bază SNMP sunt comenzi simple, ceea ce a dus la acceptarea ca protocol standard. Acestea sunt [14][15]:

- GET: o cerere de la manager la dispozitivul de controlat pentru a returna una sau mai multe valori;

- GET NEXT: similar cu GET, această operație returnează valoarea următorului OID din arborele MIB;
- GET BULK: această operație este utilizată pentru returnarea unor volume mari de date din tabelul MIB;
- SET: operația este utilizată de către manageri pentru a modifica sau asigna o valoare în dispozitivul controlat;
- TRAPS: spre deosebire de comenzile anterioare care sunt inițiate de către manager, această comandă este inițiată de către agenți, fiind un semnal către manager creat la apariția unui eveniment;
- INFORM: această comandă este similară cu TRAP, cerând în plus față de aceasta confirmarea recepționării mesajului de către manager;
- RESPONSE: aceasta este comanda utilizată pentru a returna valoarea sau semnalul acțiunilor direcționate de către managerul SNMP.

Mesajele SNMP fac parte din suita protocolului TCP/IP, acestea fiind transmise prin intermediul pachetelor UDP și apoi încapsulate și transmise în protocolul Internet. În modul standard, protocolul operează pe portul 161, iar pentru operațiile TRAP/INFORM, operează pe portul 162. Diagrama următoare ilustrează modelul bazat pe 4 nivele dezvoltat de către Departamentul de Apărare american (DoD). [14]

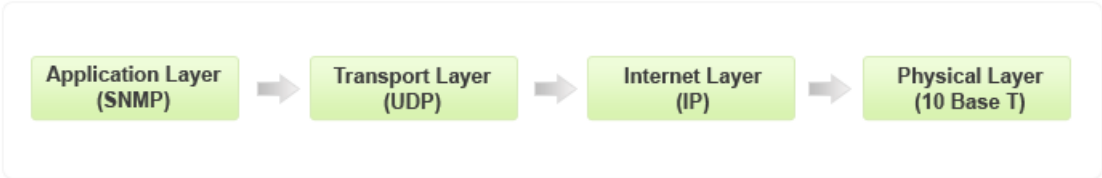

Figura 11 – Nivelele utilizate în comunicația SNMP [14]

Switch-ul, precum și celelalte echipamente care pot fi controlate prin rețea, poate opera folosind protocolul SNMP. Protocolul este implementat sub formă de agent ca aplicație pe echipament. Folosind mecanismele explicate anterior, un manager poate interoga și controla switch-ul.

Diagramele de secvență următoare exemplifică utilizarea operațiilor SNMP pentru operarea unui switch.

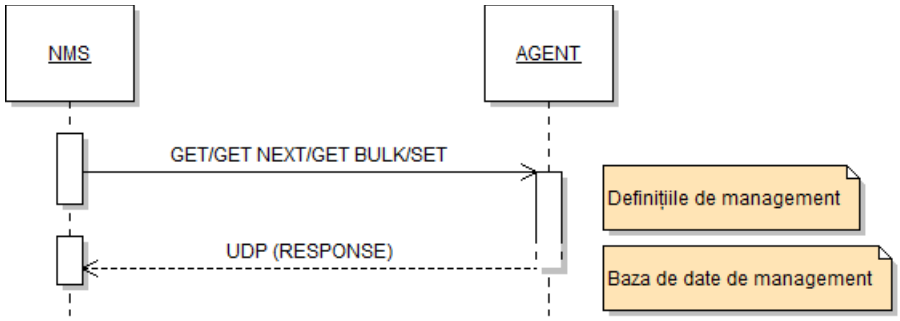

Figura 12 – Diagrama de secvență pentru operațiile de GET/GET NEXT/GET BULK/SET

NMS-ul, care poate fi un calculator la care operează un utilizator autorizat, inițiază comanda dorită către echipamentul de controlat. Spre exemplu, sub Linux se poate utiliza pachetul de programe Net-SNMP, o comandă disponibilă din acest pachet fiind următoarea [16]:

snmpget -c public zeus system.sysDescr.0

Aceasta cere agentului echipamentului interogat cu numele de rețea zeus variabila system.sysDescr.0 folosind string-ul community public.

În figurile următoare au fost exemplificate diagramele de secvență pentru operațiile TRAP și INFORM, acestea subliniind diferențele dintre cele două operații.

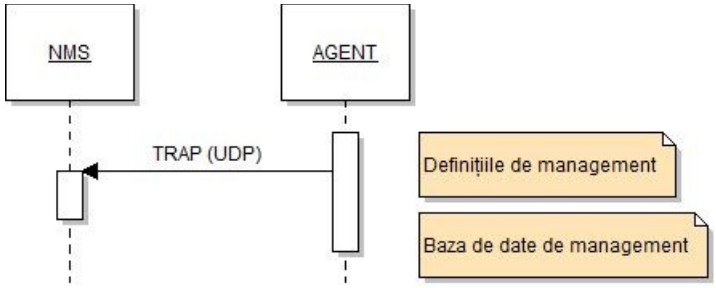

Figura 13 – Diagrama de secvență pentru operația TRAP

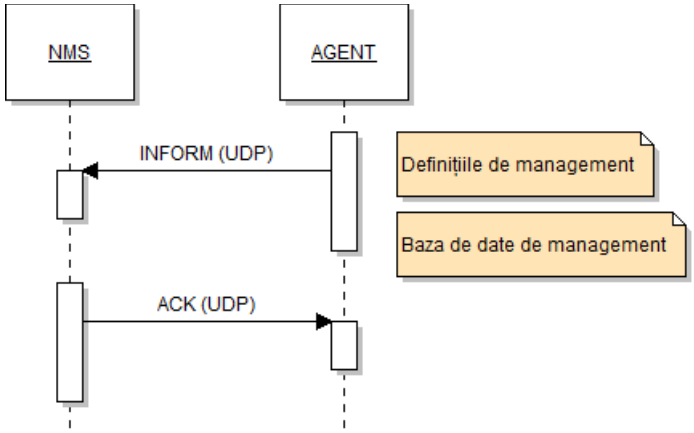

Figura 14 – Diagrama de secvență pentru operația INFORM

Bibliografie

[1] Modelul OSI 2 – http://reteledecalculatoare.webgarden.ro/menu/retele-de-calculatoare/modelul-osi-2

[2] OSI Model – http://en.wikipedia.org/wiki/OSI_model

[3] Network switch – http://en.wikipedia.org/wiki/Network_switch

[4] Network Switching Tutorial – http://www.technick.net/public/code/cp_dpage.php?aiocp_dp=guide_networking_switching

[5] VLAN Operations – http://www.informit.com/library/content.aspx?b=CCNP_Studies_Switching&seqNum=14

[6] Data link layer – http://en.wikipedia.org/wiki/Data_link_layer

[7] The Retro-Computing Society of RI, Kalpana Switch – http://www.rcsri.org/collection/kalpana/

[8] Mini GBIC Module – http://www.exprodirect.com/mini-gbic-module-1000base-sx-gigabit-lc.html

[9] InfiniBand Cable – http://www.m2cables.us.com/InfiniBand-4X-SDR-DDR-SFF-8470-Infiniband-Cable-p/m2infiniban2-main.htm

[10] RTOS – Real Time Operating Systems – http://www.engineersgarage.com/articles/rtos-real-time-operating-system

[11] Real time computing – http://en.wikipedia.org/wiki/Real-time_computing#Criteria_for_real-time_computing

[12] VxWorks – http://en.wikipedia.org/wiki/VxWorks

[13] VxWorks Kernel Programmer's Guide 6.2 – http://www.uio.no/studier/emner/matnat/fys/FYS4220/h11/undervisningsmateriale/laboppgaver-rt/vxworks_kernel_programmers_guide_6.2.pdf

[14] What is SNMP – https://www.manageengine.com/network-monitoring/what-is-snmp.html

[15] Simple Network Management Protocol – http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

[16] SNMPGET – http://www.net-snmp.org/docs/man/snmpget.html