# Comunicare intr-un Wireless Distributed System

Artiom Fașcian
Delia Popescu

7 februarie 2014

**Rezumat**

O retea ad-hoc reprezinta o colectie de noduri mobile care formeaza o retea instantanee, fara o topologie fixa. Intr-o astfel de retea, fiecare nod se comporta atat ca router cat si ca host, in mod simultan. Orice nod poate abandona reteaua sau poate restabili conexiunea in orice moment, starea celorlalte noduri ramanand neschimbata. Reteaua creata initial nu are o infrastructura de baza precum in retelele conventionale.

# 1 Introducere

Retelele bazate pe senzori wireless combina activitaile de calcul, comunicare si achizitie de date senzoriale. Utilizand protocoale mesh avansate de retea, aceste dispozitive creaza o arie de conectivitate care extinde cercetarea spatiului cibernetic in lumea fizica.

Domeniul conectivitatii retelelor mesh se afla intr-o continua cercetare si dezvoltare pentru a descoperi noi cai de comuniare prin transmitearea datelor din nod in nod, pana la punctul destinatie. Dispozitivele inteligente reprezinta urmatorul pas in automatizarea sistemelor pentru cladiri, utilitati, industrie, locuinte, transport.

Un mediu inteligent se bazeaza in primul rand pe receptionarea de date senzoriale de la mediul inconjurator. Provocarile in detectarea cantitatilor relevante, monitorizarea si colectarea de date, asignarea si evaluarea informatiilor, formulatrea de afisaje bine documentate pentru utilizatori, implementarea functiilor de alarma sunt foarte mari. Importanta retelelor bazate pe senzori este evidentiate de numarul de initiative de dezvoltare, exemplu fiind programul DARPA SENSIT, programe militare, precum si NSF Program.

Elementul principal in retelele de senzori wireless consta in capabilitatea de dezvoltare a unui numar mare de noduri mici care se asambleaza si configureaza singure. Scenariile de utilizarea a acestr tipuri de dispozitive pornesc de la monitorizarea in timp real, monitorizarea conditiilor de mediu, a mediile de calcul omniprezente, pana la monitorizarea structurii si rezistentei echipamentelor.

Pe langa reducerea drastica a costurilor, retelele de senzori wireless posed abilitatea de a se adapta unui mediu inconjurator in schimbare. Mecanismele de adaptare pot raspunde la schimbarie in topologiile de retea or pot cauza o schimbare in modul de operare al retelei.

Actualmente retelele wireless se afla la inceputul dezvoltarii, atingand o parte mica din problemele emergente ale integrarii comunicatiei low-power, de detectare senzoriala, de stocare a energiei si de calcul. Spre deosebire de sistemele fara fir traditionale, nodurile de senzori wireless nu trebuie sa comunice direct cu cel mai apropiat turn de control sau statie de baza, ci doar cu nodurile vecine.

Viziunea retelelor mesh se bazeaza pe puterea numerelor. Spre deosebire de sistemele de telefonie mobila, care pierd semnalul cand mai multe dispozitive se siteaza intr-un perimetru mic, interconecatarea retelelor wireless creste pe masura ce creste numarul de noduri din retea. Atat timp cat exista o densitate suficienta, o singura retea de noduri poate acoperi zone nelimitate.Cu fiecare nod care comunica pe o raza de de metri si costa mai putin de 1 dolar, o retea de senzori care inconjoara ecuatorul pamantului costa mai putin de 1 milion de dolari.

# 2 Hardware

ATMega128RFA1 este un microcontroller compatibil IEEE802.15.4, single-chip, care combina tehnologia microcontroller-ului AVR si celui mai bun tranceiver de 2.4GHz RF, oferind cea mai buna performanta in industria RF pentru despozitivele single-chip. Caracteristicile ATMega128RFA1 includ Wake-on Radio, numarator de 32 de biti pentru simbol MAC, senzor de temperatura, moduri de transmisie automata, motor de criptare AES pe 128 de biti, generator de numere aleatoare,module pentru rate mari de date, suport pentru diverse antene.

Tranceiver-ul de pe microcontrollerul ATMega128RFA1 permite acestuia sa transmita in diferite benzi din 2.4GHz. Permite viteze de transmisie de 250kbps, 1Mbps si 2Mbps. Are doua moduri de functionare, Basic si Extended. Modul Extended include multe functionalitati, prevazute de obicei in nivelurile de acces la mediu. In acest caz, functionalitatile sunt cele necesare pentru implementarea IEEE 802.15.4.

Interfata cu microcontrollerul este necesara pentru comandarea tranceiverului din program si pentru a obtine/trimite date catre acesta. RFA1 pune la dispozitie urmatoare interfata:

- Registrii de comanda - Sunt tratati ca orice alt registru I/O si sunt accesati la fel de rapid. Sunt folositi pentru a transmite comenzi catre tanceiver sau pentru a obtine informatii punctuale despre functionarea acestuia. Exemple: $TRX\_STATE, TRX\_STATUS, PHY\_RSSI$. Folosirea acestor registri este posibila doar atunci cand ceasul ceasul tranceiver-ului nu este oprit.

- Frame buffer - Este o zona de memorie de 128 de octeti, in care se recetioneaza/ din care se transmit pachete. Este accesibila din microcontroller tot ca o zona de registri I/O de 128 de octeti, inceputul este TRXFBST si sfarsitul este TRXFBEND. Scrierea/citirea in/din frame-buffer mereu dureaza un ciclu de ceas, adica un maxim de 128 Mbps.

- Registrul TRXPR - Mostenire din timpurile in care microcontrollerul si tranceiverul erau in chipuri separate, TRXPR este registrul care detine doua flag-uri (in trecut doi pini) care afecteaza tranceiverul chiar daca acesta este in sleep (bitul TRXRST poate reseta oricand tranceiverul, bitul SLPTR controleaza atat starea de sleep a tranceiverului cat si startul transmisiei unui pachet).

- Intreruperi - Tranceiverul poate fi configurat sa genereze mai multe intreruperi microcontrollerului (exemplu: $TRX24\_RX\_START$, care alerteaza citirea imediata a frame bufferului dupa inceperea receptiei unui pachet; $TRX24\_TX\_END$, care semnaleaza sfarsitul transmisiei).

Nodul senzorial Sparrow este echipat cu trei tipuri de senzori: senzor de temperatura, de umiditate si de luminozitate ambientala. Senzorul de luminozitate este analogic si furnizeaza o tensiune direct proportionala cu nivelul iluminarii ambientale. Tensiunea poate fi citita de catre microcontrollerul ATMega128RFA1 de pe portul F, pinul2 (PF2). Tot pe portul F, pinul PF0 se citeste si tensiunea de alimentare a nodului senzorial. Valoarea tensiunii citite este divizata cu 2. Senzorul de umiditate si temperatura este de tipul SHT si este conectat la interfata I2C a microcontroller-ului.

Pentru a retine modul de operare curent, tranceiver-ul de pe RFA1 foloseste o masina de stari (daca transmite, receptioneaza, daca este in standby, daca se pregateste sa transmita etc). Starile aferente sunt:

- SLEEP: Tranceiverul se afla in stare de asteptare, nu consuma curent;
- RESET: Tranceiver-ul tocmai a fost resetat;
- $TRX\_OFF$: Starea de standby, in care nu se asculta mediul pentru pachete;
- $RX\_ON$ Starea in care se asculta mediul;
- $PULL\_ON$ Starea in care componenta de transmisie este activa si pregatita;
- $BUSY\_TX$ Se transmite un pachet;

  Tranzitiile intre stari sunt fie declansate de evenimente exterioare, si atunci sunt subliniate cu verde, fie prin comanda de la microcontroller (iar atunci sunt colorate cu albastru sau cu rosu). Registrii $TRX\_STATE$ si $TRX\_STATUS$ se ocupa de controlul si observarea masinii de stari.

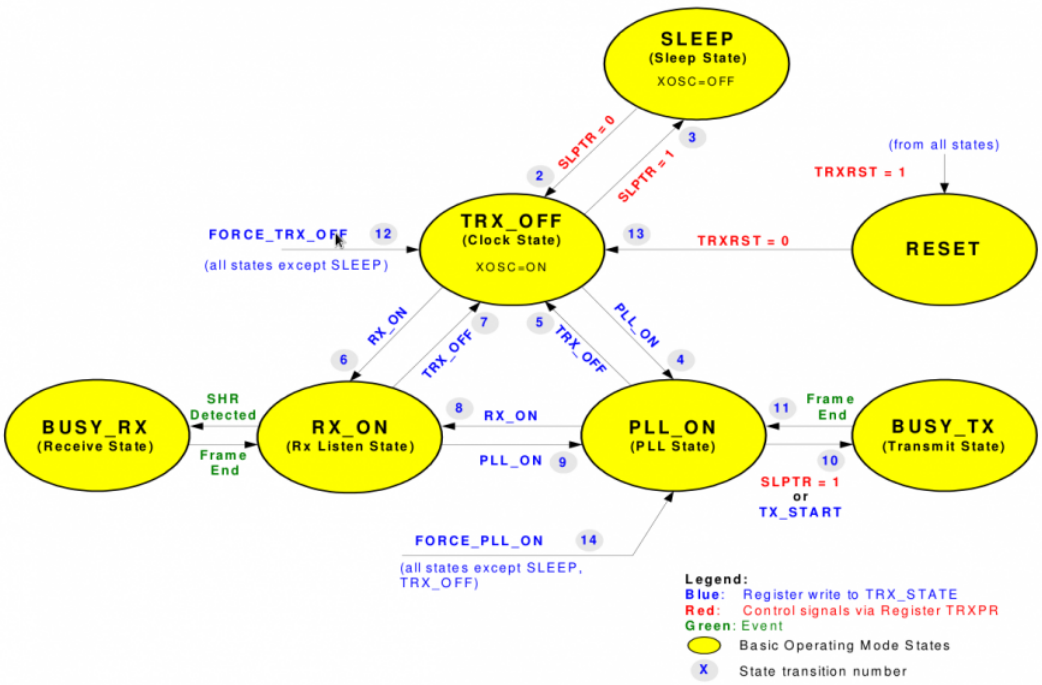


Figura 1: Masina de stari

# 3 Comunicatia

Protocolul de baza folosit in implementarea proiectului este Destination-Sequenced Distance Vector (DSDV). Protocolul DSDV este o modificare a protocolului de rutare conventional Bellman-Ford. Acesta abordeaza dezavantajele legate de proprietatile slabe ale protocolului de rutre RIP (Routing Information Protocol), in ceea ce priveste legaturile de retea intrerupte pentru retele de rutare ad-hoc. DSDV adauga un atribut nou, un sequence number, pentru fiecare intrare din tabela de rutare pentru RIP-ul conventional. Utilizand cel mai recent adaugat sequence number, nodurile mobile pot obtine informatii referitoare informatiile de rutare de la nodurile noi, prevenind astfel formarea loop-urilor de rutare.

In DSDV fiecare nod mobil dintr-o retea ad-hoc mentine o tabela de rutare, care listeaza toate destinatiile disponibile, metrica si next-hop-ul pentru fiecare destinatie si sequence number gederat de nodul destinatie. Utilizand o astfel de tabela de rutare, stocata in fiecare nod, pachetele sunt transmise direct intre nodurile din retea. Fiecare nod dintr-o retea ad-hoc updateaza tabela de rutare periodic sau cand este disponibila o informatie noua, pentru a mentine consistenta tabelei de rutare, tinand cont de schimbarea dinamica a topologiei retelei.

Periodic, sau imediat dupa ce au loc schimbari in topologia retelei, ficare nod mobil anunta informatiile de rutare prin transmiterea unui pachet cu informatiile updatate din tabela de rutare, prin broadcast sau multicast. Fiecare pachet de update incepe cu metri egal cu 1, pentru a directiona nodurile conectate. Astfel se indica faptul ca fiecare vecin care primeste date este la un hop distanta (metric= 1) de nod. Dupa receptionarea pachetului, vecinii updateaza tabela de rutare prin incrementareavaloarea metrici cu 1 (next hop) si transmit pachetul updatat catre vecinii fiecaruia dintre noduri. Procesul se va repeta pana cand toate nodurile din retea au primt un pachet cu noua tabela de rutare si valoarea corespondenta pentru metric. Datele de update sunt de asemenea mentinute pentru o perioade de timp, pentru a determina cea mai buna ruta catre fiecare nod destinatie, apoi updateaza tabela de rutare si retransmite noul pachet in retea. Daca un nod primeste mai multe pachete de update pentru acceasi destinatie in timpul perioadei de asteptare, rutele cu cel mai recent sequence number sunt preferate. Insa, daca nu s-a schimbat decat sequence number, informatiile de update nu sunt mereu transmise imediat. Daca pachetul de update are acelasi sequence number al aceluiasi nod, va fi folosit pachetul de update cu cea mai mica metrica, iar ruta existenta va fi anulata sau stocata ca rutamai putin preferata. Anuntarea rutelor care sunt pe punctul de a se schimba poate fi intarziata pana cand se detecteaza cele mai bune rute.

Elementele din tabela de rutare pentru fiecare nod se schimba dinamic, pentru a pastra compatibilitatea cu schimbarile in topologia retelei. Pentru a pastra consistenta, anunturile referitoare la datele din tabela de rutare trebuie sa fie frecvente sau suficient de rapide pentru a se asigura ca fiecare nod din retea este in masura, in orice moment, sa localizeze toate celelalte noduri din retea.

Intr-un proces de update al tabelei de rutare, nodul original eticheteaza fiecare pachet de update cu un sequence number, pentru a distinge updateurile unul de celalalt. Sequence number este un numar monoton crescator, care identifica in mod unic fiecare update de la un nod dat. Ca rezultat, daca un nod primeste un pachet de update de la un alt nod, sequence number trebuie

sa fie cel pun egal cu sequence number-ul nodului corespondent din tabela de rutare. Altfel noua informatie de rutare receptionata nu mai este valida si trebuie aruncata.

Prin modificarile aduse, DSDV constituie un protocol de rutare mai adecvat pentru retelele ad-hoc.

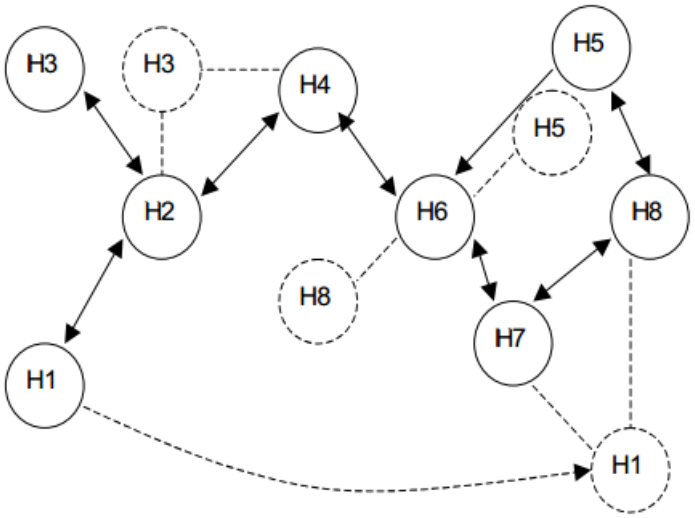


Figura 2: Exemplu de retea ad-hoc

# 4 Implementare

Poiectul a fost implementat pe principiul protocolului DSDV, folosind aceleasi reguli generale de implementare. Aplicatia realizata are capabilitatea de a fi scalata la un numar de aproximativ 1000 de noduri.

La nivelul fiecarui nod din retea, pentru receptionare de mesaje, transmiterea de mesaje si stocarea tabelei de rutare este implementata o stiva, care lucreaza pe principiul initializare, push, pop, empty.

Pentru aceasta, in codul de mai jos este definit un vector care stocheza datele de receptie, respectiv transmisie de date: $uint8\_tbuffer[128]$ ; $booltoSend$, pentru receptie sau transmisie (poate fi de doua tipuri : $TO\_SEND$ sau $TO\_RECEIVE$); $intmaxsize$ pentru a stoca dimensiunea maxima pentru structuri (dimensiunea stivei) de tip $msg\_t$; $msg\_t * contents$, pentru a stoca maxsize, definit anterior, pentru structuri de tip $msg\_t$; $int\_top$ care pointeaza catre varful stivei; functia $voidinitStack(stack\_t * stack, intstackSize)$ care initializeaza stiva (parametrii definesc pointerul catre stiva si dimensiunea acesteia); functia $voidpushStack(stack\_t * stack, msg\_tmessage)$ care adauga un element nou in stiva; $msg\_tpopStack(stack\_tstack)$; pentru a extrage un element din stiva; $intisEmptyStack(stack\_t * stack)$; care daca stiva este goala, resturneaza numarul curent de elemente din stiva; $intisFullStack(stack\_tstack)$; care verifica daca stiva a ajuns la capacitatea maxima de stocare.

Listing 1: Mesaje

```
1  #define TO_SEND 1
2  #define TO_RECV 0
3
4  typedef struct{
5  uint8_t buffer[128];
6  bool toSend;
7  } msg_t;
8
9  typedef struct {
10   msg_t *contents;
11   int maxSize;
12   int top;
13 } stack_t;
14
15 void initStack(stack_t *stack, int stackSize);
16 void pushStack(stack_t *stack, msg_t message);
17 msg_t popStack(stack_t *stack);
18 int isEmptyStack(stack_t *stack);
19 int isFullStack(stack_t *stack);
```

Pentru procesul de initializare a tabelei de rutare, adaugarea unui nod in retea, cautarea unei intrari,respectiv stergere unei intrari am introdus o lista simplu inlantuita. Pentru aceasta am definit urmatoarele: $uint8\_tdestinationAddress$ pentru stocarea adresei destinatie; $uint8\_tnextHop$ pentru definirea nodului din imediata apropiere care stie de adresa destinatie; $uint8\_thopCount$ care defineste numarul de noduri/hop-uri pana la nodul destinatie; $routingEntry\_t$, structura pentru o intrare din tabela de rutare; $routingEntry\_tentry$ defineste o intrare in tabela de rutare; $structroutingTableNode *nextNode$ defineste nodul care constituie urmatoarea intrare in tabela; $routingTableNode\_t$, element din lista ce reprezinta tabela de rutare; $routingTableNode_tinitRoutingTable(routingEntry\_tvalue)$ realizeaza procesul de initializare a tabelei de rutare; $routingTableNode\_t \times addRoutingEntry(routingEntry\_tvalue, booladd\_to\_end)$ , adauga un element nou in tabela de rutare; $routingTableNode\_t *searchRoutingEntry(routingEntry\_tvalue)$ realizeaza cautartea unei valori in tabela de rutare, folosita pentru detectarea dublurilor din tabela; $routingTableNode\_t *deleteRoutingEntry(routingEntry\_tvalue)$, folosit in momentul disparitiei unei rute, pentru stergerea acesteia din tabela de rutare; $routingTableNode\_tsearchRoutingEntryDestination(uint8\_tdestination)$ utilizat pentru identificarea intrarii optime in tabela de rutare, este implementat pentru comunicatia intre nivelul de aplicatie si rutarea efectiva.

Listing 2: Lista simplu inlantuita

```
1  typedef struct {
2          uint8_t destinationAddress;
3          uint8_t nextHop;
4          uint8_t hopCount;
5  } routingEntry_t;
6  typedef struct routingTableNode{
7          routingEntry_t entry;
8          struct routingTableNode *nextNode;
9  } routingTableNode_t;
10
11
12
13 routingTableNode_t *initRoutingTable(routingEntry_t value);
14 routingTableNode_t *addRoutingEntry(routingEntry_t value, bool
       add_to_end);
15 routingTableNode_t *searchRoutingEntry(routingEntry_t value);
16 routingTableNode_t *deleteRoutingEntry(routingEntry_t value);
17 routingTableNode_t *searchRoutingEntryDestination(uint8_t
       destination);
```

# 5 Exemplu cu un numar de 4 noduri

Primul pas in implementarea retelei constituie asignarea unui identificator unic pentru fiecare nod in parte, dupa modelul DHCP.
In modelul de implementare am folosit o valoare "max" care stocheaza numarul maxim de identificare al ultimului nod din retea.

- Pentru autentificarea nodului 1 in retea, acesta trimite un pachet de broadcast "Hello", care consta intr-o cerere de conexiune in retea. Fiind primul nod, nu va primi replay la mesaj si isi asigneaza valoarea max=1.
- Nodul 2 trimite cerere de autentificare, primita doar de nodul 1, care ii trimite replay cu valoarea max=1. Nodul 2 asigneaza in tablea de rutare valoarea pentru max=2 si trimite mesaj de broadcast noua valoare pentru max .
- Nodul 3 trimite mesaj de broadcast , "Hello", care ajunge la nodurile 1 si 2. Acestea raspund cu valoarea pentru max=2, nodul 3 trimite mesaj broadcast catre nodurile din retea cu max=3.
- Nodul 4 trimite mesaj de broadcast,"Hello", care ajunge la nodurile 1,2,3. Acestea trimit inapoi mesaj cu valoarea curenta pentru max=3. Nodul 4 receptioneaza mesajul, updateaza valoarea max la 4 si trimite mesaj de broadcast in care max=4.

In Figura 3 este ilustrat procesul de stabilire a conexiunii prezentat mai sus.

In Figura 4 sunt ilustrate mesajele transmise pentru sincronizarea valorii max.

In cazul in care un nod din retea devine indisponibil( nodul 1 in Figura 5), pentru reconectarea acestuia se reia procesul ilustrat in figurile 3 si 4. Nodul 1 trimite un mesaj de broadcast "Hello " catre toate celelalte noduri din retea. Primeste replay cu in care valoarea max=4 , actualizeaza max=5 si trimite mesaj pentru sincronizarea noii valori.

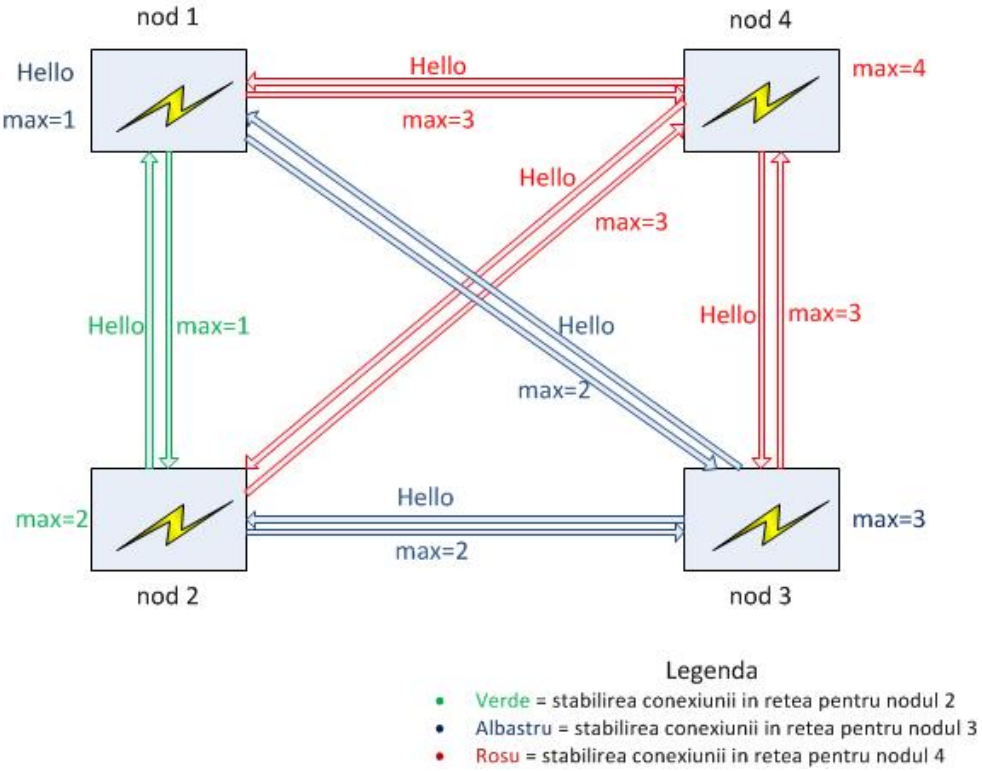


Figura 3: Stabilirea legaturii in retea

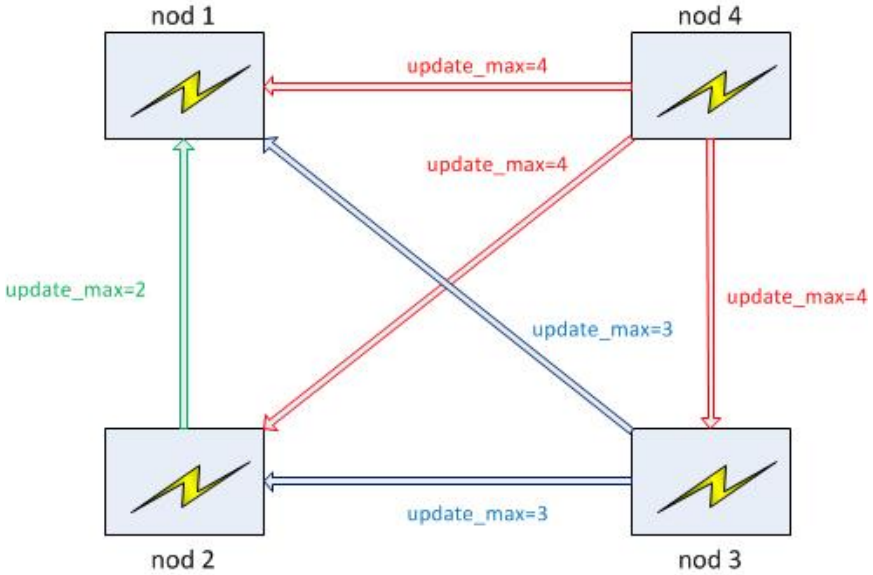


Figura 4: Sincronizare

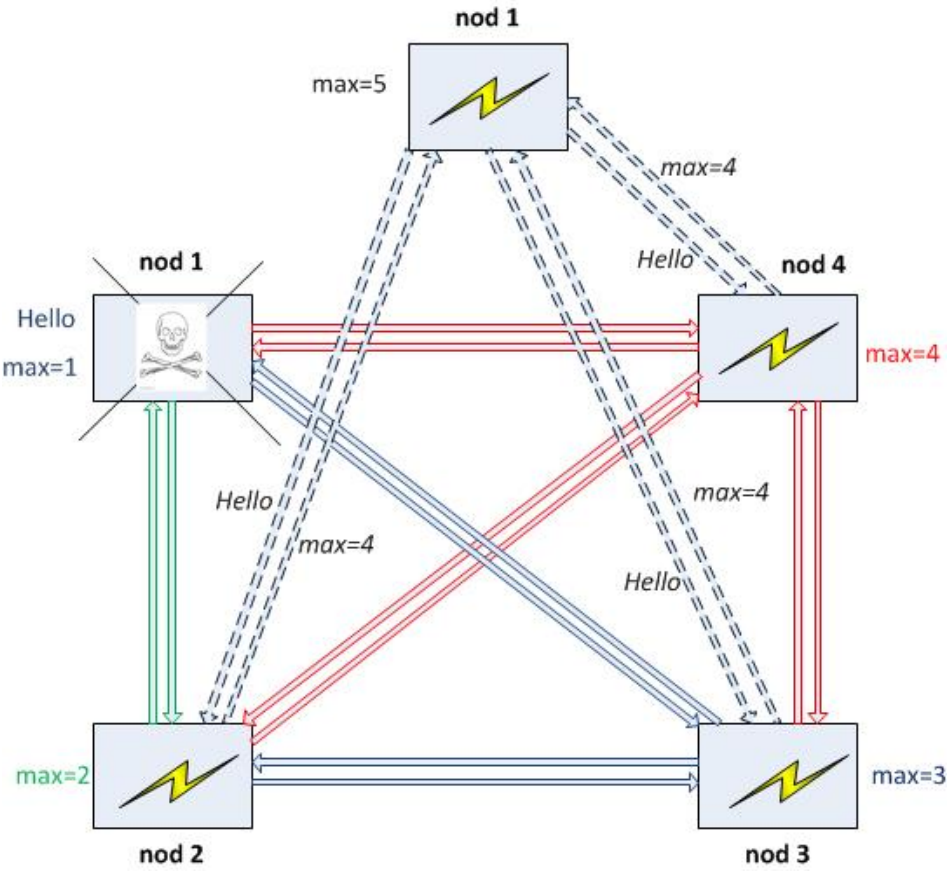


Figura 5: Stabilirea legaturii in retea dupa deconectarea unui nod

# 6 Tabela de rutare

Tabela de rutare contine 4 intrari: destinatie, next hop, TTL si numarul de hop-uri pana la destinatie. Fiecare nod din retea contine propria tabela de rutare. Transmisia pachetelor in retea de la un nod sursa la un nod destinatie se realizeaza in conformitate cu datele din tablea de rutare, pentru a se folosi calea cea mai scurta. Optiunea TTL va fi discutata in sectiunea "Update pentru tabela de rutare".

Dupa procesul de asignare a unui identificator pentru un nod nou din retea, nodurile deja existente, care au receptionat mesajul de broadcast cu noua valoare pentru max, trimit mesaj catre nodul nou cu tablela de rutare. Noul nod receptioneaza primul mesaj cu tabela de rutare si stocheaza informatiile in stiva. Pentru toate mesajele receptionate ulterior, secvential, verifica datele din tabela pentru a detecta eventualele modificari, respectiv o tabela de rutare noua. Daca informatiile nu difera pastreaza prima tabela de rutare, in caz contrar arunca prima tabela si stocheaza datele receptionate ulterior. Next-hop-ul din tabela de rutare se calculeaza conform protocolului DSDV, incrementand valoarea unui metric de catre fiecare nod in parte, la receptionarea tabelei.

Fata de protocuId DSDV, in aplicatia implementata nu se realizeaza update-

ul tabelei de rutare.

Toata procesarea este efectuata in fisierul routing.c si header-ul respectiv routing.h.

Listing 3: Functia de procesare a mesajelor de rutare

```
1  void processStack(void)
2  {
3  msg_t newMessage;
4  routingEntry_t entry;
5  while ( isEmptyStack(&msgQueue) > 0 )
6  {
7    newMessage=popStack(&msgQueue);
8
9
10   if (newMessage.toSend==TO_SEND)
11     sendFrame(newMessage.buffer);
12
13   if (newMessage.toSend==TO_RECV)
14   {
15
16     switch (newMessage.buffer[1])
17     {
18       case ROUTE_UPDATE:
19         //updateRouting();
20         break;
21
22       case DATA:
23         //processData();
24         break;
25
26       case HELLO:
27         sendMax(MAX_VALUE);
28         break;
29
30       case MAX_VALUE:
31         if (node_address==0)
32         {
33           max=newMessage.buffer[2];
34           max++;
35           node_address=max;
36           sendMax(UPDATE_MAX);
37           }
38         break;
39
40       case UPDATE_MAX:
41         max=newMessage.buffer[2];
42         break;
43
44       case ROUTE_SEND:
45         entry.destinationAddress=newMessage.buffer[2];
46         entry.nextHop=newMessage.buffer[3];
47         entry.hopCount=newMessage.buffer[4]+1;
48         addRoutingEntry(entry,1);
49         break;
50
51
52       case SCREAM:
53         if (node_address!=0)
54           oneMessageSend(SHUTUP);
55
56         break;
```

```
57          case SHUTUP:
58            if (node_address!=0)
59            {
60              sendRouting();
61              state=1;
62            }
63            break;
64
65        }
66      }
67  }
68  }
```

Functia *processStack()* se ocupa cu managementul de date si tabelei de rutare cit si a protocolului initial de discutie. Intr-o bucla este efectuata extragerea datelor din stiva specifica fiecarui nod. In dependenta de flagul *toRecv* mesajul este procesat de nod sau transmis. Pentru partea de transmitere se construieste un frame din buferul mesajului care se paseaza catre registii hardware ale microcontrolerului.

Daca mesajul este destinat procesarii nodului, in dependenta de tipul acestuia se fac diferite operatii: transmiterea identificatorului pentru nod, transmiterea rutelor, ideea de wake-up.

# 7 Sleep si Wake-up

Fiind necesara sincronizarea nodurilor in reteaua ad-hoc, de altfel si reducerea consumului, perioada de sleep a unui nod necesita o abordare speciala. Initial au existat 2 solutii:

- Trezirea tuturor nodurilor la un moment dat sau, prin alte cuvinte, creearea unei bariere de sincronizare care ar pornirea tuturor nodurilor dintr-un punct si schimbul de mesaje pe perioada activa a acestora. Problema cu aceasta abordare este inaptitudinea unui nod de a deosebi vecinul sau daca acesta este in sleep sau nu mai exista in retea.
- Trezirea pe rind a nodurilor si comunicarea pe rind a mesajelor intre noduri.

Intr-un final a fost aleasa a doua abordare aceasta fiind prezentata mai jos.

Incazul procesului de trezire a nodurilor din starea de sleep, se procedeaza dupa cum urmeaza:

Se considera toate nodurile aflate in starea de sleep.

- In momentul in care un nod (exemplu nodul 1 in Figura 6) doreste sa transmita un mesaj la un alt nod, destinatie, (nodul 4 in exemplul dat) trimite pachete de tip broadcast cu apel de wake up.
- Primul nod care a receptionat pachetul de broadcast (nodul 2 in Figura 6) trece in starea de wake up si transmite mesaj de tip sleep catre nodul de la care a primit apel (nodul 1). Nodul initiator, dupa receptionarea mesajului, trece in starea sleep.
- Nodul care a ramas treaz trimite pachete de de tip broadcast cu mesajul wake up catre nodurile vecine, dupa care procesul continua pana la nodul destinatie.

- In orice moment exista un nod treaz care trimite mesaj de broadcast sa trezeazca nodurile vecine pentru perpetuarea mesajului pana la nodul destinatie.
- La capatul celalalt, nodul destinatie, dupa receptionarea mesajului,trimite mesaj de broadcast catre vecini pentru a anunta nodul sursa (care acum a devenit destinatie) de receptionarea mesajului. Prin urmare, procesul detaliat anterior, se repeta in sens invers pentru acknowledge.
- Tabela de rutare este transmisa secvential, de catre nodurile aflate in starea wake, incepand de la nodul initiator, dupa transmiterea mesajului de wake up.

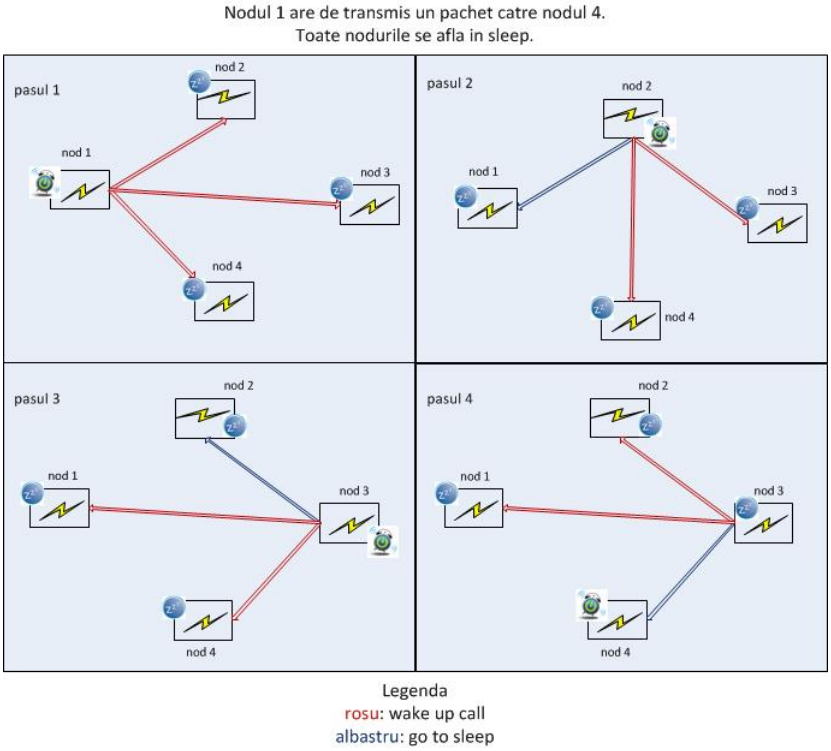


Figura 6: procesul de sleep si wake up

# 8 Update pentru tabela de rutare

Pentru a intelege daca un mesaj ar putea ajunge catre un nod, tabela de rutare necesita o actualizare. Problema aparenta in designul propus este deosebirea nodurilor care sunt in starea sleep de cei care sunt deconectati de la retea. In acest fel se poate considera doar timpul in care un nod este treaz.

Sa presupunem o retea cu 4 noduri prezentata in figura 6. Fiecare nod, pentru fiecare intrate din tabela de rutare are cate un TTL. Variabila data este responsabila pentru a diferentia un nod viu de unul ce e in starea sleep.

De fiecare data cand un nod se trezeste acesta decrementeaza aceasta valoare. In designul propus, valoarea acesteia este 255 (maximum pe un octet). Daca destinatia unei intrari din tabela de rutare coincide cu adresa nodului, aceasta valoare nu se decrementeaza. La transmiterea tabelei de rutare, valoarea TTL se compara cu intrarile individuale din tabela. Daca TTL-ul este mai mare atunci se updateaza intrarea cu noile valori. In caz in care aceasta valoare ajunge la 0, intrarea din tabela este stearsa.

# 9 Feature development

Proiectul prezentat prezinta doar baza unei stive de comunicare intre o retea de noduri wireless. Exista un sir de imbunatatiri legate de dezvoltarea acestei stive. Desigur, pentru a obtine calitate a comunicatiei se poate implementa stiva inspirindune din standartele RFC pentru retelistica.

Stiva de comunicare prezentata initial nu a avut ca arhitectura delimitarea clara a aplicatiei unui utilizator de partea de logistica a traficului. O directie dorita ar fi delimitarea clara a nivelelor de retea si aplicatie.

# 10 Concluzii

In cadrul acestei lucrari a fost pusa baza unei stive de comunicatie. Cercetarea pe parcursul implementarii acestui proiect a ajutat la intelegerea mai buna ce inseamna o comunicatie intr-o retea ad-hoc.

# Bibliografie


[1] Jason Lester Hill, System Architecture for Wireless Sensor Networks, in , Spring 2003, 1-5 .

[2] Guoyou He , Network Laboratory, Destination-Sequenced Distance Vector (DSDV) Protocol *DSDV Protocol*, Helsinki University of Technology).

[3] *DSDV documentation*, `http://www.cs.jhu.edu/~cs647/dsdv.pdf`

[4] *C Linked List Data Structure Explained with an Example C Program*, `http://www.thegeekstuff.com/2012/08/c-linked-list-example/`

[5] Wii, *elf.cs.pub.ro*, `http://elf.cs.pub.ro/wsn/wiki/lab2i`

[6] Wii, *Wikipedia Encyclopedia*, `http://en.wikipedia.org/wiki/Wii`