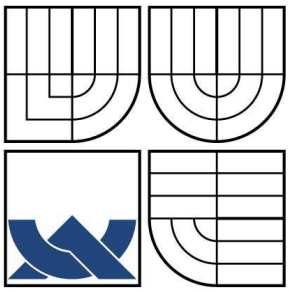
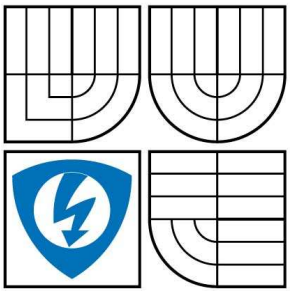


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

HROMADNÁ KOMUNIKACE V BEZDRÁTOVÝCH SENZOROVÝCH SÍTÍCH

MULTICAST IN WIRELESS SENSOR NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. JAROMÍR VALOUCH

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MILAN ŠIMEK

BRNO 2009

ANOTACE

Diplomová práce se zaobírá výzkumem v oblasti spotřeby energie v bezdrátových senzorových sítích. Je zaměřena na spotřebu energie při komunikaci a způsobech minimalizace této spotřeby. V úvodu jsou objasněny základní vlastnosti senzorových sítí, jejich rozdělení v závislosti na způsobu použití a používané komunikační technologie. Dále jsou zde vysvětleny pojmy hromadná komunikace, její výhody, rozdíl mezi unicast a multicast komunikací, způsoby tvorby skupin při multicast komunikaci, adresace těchto skupin a používané adresy, nároky na směrovače při multicast komunikaci. Jsou zde popsány multicastové směrovací protokoly, techniky směrování, tvorby směrovacích tabulek, zabránění nadbytečného přeposílání paketů, rozdíly v tvorbě multicast skupin. V další kapitole je popsán IGMP protokol, který slouží k výměně informací o členství uživatelů v multicast skupinách, jsou zde popsány formáty IGMP zpráv, rozdíly protokolu verze 1, verze 2 a verze 3. Dále je vysvětlen pojem ad-hoc sítě popsány výhody použití těchto sítí, ad-hoc směrovací protokoly unicast AODV a multicast MAODV, formáty zpráv obou směrovacích protokolů, jakým způsobem dochází k výměně těchto zpráv. Je zde rozebrána architektura prvku v bezdrátové senzorové síti se zaměřením na komunikační obvody a spotřebu jejich energie. Ke konci práce jsou uvedeny výsledky simulace, kde byla zkoumána spotřeba energie jednotlivých prvků bezdrátové senzorové sítě v závislosti na způsobech přenosu. Pro simulování komunikace v bezdrátové senzorové síti byl použit network simulator2, který umožňoval konfiguraci energetického modelu. Dále zde najdeme popis simulátoru network simulator2 a postup při konfiguraci tohoto simulátoru.

Klíčová slova:

Bezdrátové senzorové sítě
Multicast
Unicast
Energie
Komunikace
Simulace

ABSTRACT

This master's thesis describes energy consumption in wireless sensor networks. It is focused on energy consumption during communication and it provides rules to save energy during this operation. There are terms like wireless sensor networks, multicast, ad-hoc networks, routing protocols, multicast routing protocols, IGMP protocol, AODV routing protocol, MAODV routing protocol explained in this thesis. The main difference between multicast and unicast communication is analyzed as well. In the end of this thesis there are results of simulation small wireless sensor network from network simulator2. The simulations were focused on energy consumption during communication. The length of packet was changed during communication between two nodes in this simulation. In wireless sensors network nodes play a dual role as both data sender and data router therefore there were made research into energy consumption senders, routers, receivers in wireless sensor network during communication.

Keywords:

Wireless sensor networks
Multicast
Unicast
Energy consumption
Communication
Simulation

Bibliografická citace mé práce:

VALOUCH, J. *Hromadná komunikace v bezdrátových senzorových sítích.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 55 s. Vedoucí diplomové práce Ing. Milan Šimek.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma Hromadná komunikace v bezdrátových senzorových sítích jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.“

V Brně dne ……. …….

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Milanu Šimkovi, za poskytnutí informací, podkladů a za pomoc při zpracování diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 Úvod

Bezdrátové sítě se stávají v dnešní době čím dál více populárnější, může za to zejména mobilita, kterou poskytují koncovým uživatelům. Při budování bezdrátových senzorových sítí odpadá nutnost zřizování přípojek pro každé koncové zařízení. Musíme pouze zřídit několik přístupových bodů (dle rozsahu sítě), jenž budou pokrývat oblast, v které se koncová zařízení nacházejí. Toto je zejména výhodné v případě, nachází-li se velké množství uživatelů na relativně malé ploše a bylo by téměř nemožné a neekonomické ke každému zřizovat přípojku např.bezdrátové senzorové sítě.

Pro napájení koncových zařízení v bezdrátových senzorových sítích se používají baterie, kvůli již dříve zmiňované mobilitě. Bohužel při použití baterií musíme řešit problém s jejich životností a to zejména v aplikacích, kde vyžadujeme velkou životnost (několik měsíců až roky). Životnost baterií lze zvyšovat zvětšováním kapacity baterií nebo snižováním spotřeby energie koncových zařízení. Při zvyšování kapacity baterií se úměrně zvyšuje také jejich velikost a zvětšování velikosti baterií způsobuje růst rozměrů jednotlivých prvků senzorové sítě, což není vždy žádoucí, způsobuje to problém v aplikacích, kde požadujeme malé rozměry koncových zařízení, tady se musíme vydat cestou snižování spotřeby energie u koncových zařízení.

Nejvíce energie v bezdrátových sítích se spotřebovává při komunikaci, proto se při šetření energie budeme zaměřovat zejména tímto směrem. Šetření energie docílíme v značné míře omezením maximálního dosahu koncových zařízení, čím menší dosah budou mít, tím více energie ušetříme, proto se jeví dobrým řešením koncipovat komunikaci jako multihop, což znamená, že koncová zařízení budou mít dosah jenom ke svým sousedům a ti zase ke svým atd. Komunikace bude probíhat přeposíláním paketů od jednoho uzlu ke druhém, čímž dojde k doručení paketu na požadovanou adresu.

2 Bezdrátové senzorové sítě

Bezdrátové senzorové sítě jsou označovány anglickou zkratkou WSN (wireless sensor network). Tyto sítě jsou složeny z autonomních prvků, které používají senzory pro monitorování určitých podmínek v prostředí např. měření teploty, tlaku, hluku, znečištění, snímání pohybu, detekce kouře atd. WSN může obsahovat stovky až tisíce prvků. Každý prvek bezdrátové senzorové sítě je vybavený bezdrátovým vysílačem a přijímačem, mikroprocesorem, baterií nebo popř. jiným napájecím zdrojem. Velikost jednotlivých prvků sítě je závislá od našich požadavků na prvky (např. životnost baterie, velikost paměti, dosah bezdrátového přijímače a vysílače, rychlost mikroprocesoru) a od ceny, kterou jsem ochotni zaplatit.

V oblasti, kde potřebujeme snímat nebo zaznamenávat určitou událost, jednoduše rozmístíme jednotlivé koncové prvky WSN (senzory). Pokud senzory zaznamenají určitou událost musí ji nějakým způsobem ohlásit, k tomuto účelu slouží tzv. řídící prvek sítě, který bývá výkonnější než ostatní prvky sítě, aby byl schopný zpracovat a vyhodnotit zprávy přijímané od jednotlivých senzorů. Jednotlivé senzory tudíž nemusí disponovat velkým výpočetním výkonem, čímž se snižuje jejich energetická náročnost a samozřejmě cena. Přičemž řídící prvek slouží také jako prostředník mezi WSN sítí a internetem, může tudíž po vyhodnocení určité události vyslat oznámení na předem nastavenou adresu v podobě zprávy.
Podle způsobu použití můžeme WSN sítě obecně rozdělit na WSN pro sběr dat, WSN pro sledování pohyblivých objektů, WSN pro bezpečnostní monitoring a WSN hybridní. V dnešní době jsou senzorové sítě terčem výzkumů.

2.1 Bezdrátové senzorové sítě pro sběr dat

Senzory jsou rozmístěné v prostředí, ze kterého potřebujeme získávat určité informace v závislosti na čase. Každou předem danou časovou periodu dojde k odeslání aktuálních informací ze senzorů směrem k řídícímu prvku sítě. Řídící prvek slouží buď jako sběrna těchto informací nebo jako brána, která dále přeposílá data sítí k příjemci, popř. může zastávat obě funkce. Tento sběr dat má uplatnění zejména ve vědeckých výzkumech, kdy můžeme mít rozmístěny desítky i stovky senzorů v určitě oblasti, ze kterých můžeme následně získané informace vyhodnocovat. Sběr dat může trvat několik měsíců až roků v závislosti na napájecích zdrojích a spotřebě energie jednotlivých senzorů. Jeden z hlavních požadavků na

prvky těchto sítí je především velká životnost baterie, proto většina prvků v síti setrvává ve stavu spánku a probudí se za předem určenou časovou periodu za účelem odeslání nebo příjmu dat. Naopak u těchto sítí není požadována velká přenosová rychlost.

2.2 Bezdrátové senzorové sítě pro sledování pohyblivých objektů

Jedná se o sledování určitého objektu, který se pohybuje v dosahu WSN sítě, může se např. jednat o sledování zaměstnanců v pracovním prostředí. Princip sledování je poměrně jednoduchý, sledovaný objekt musíme vybavit aktivním prvkem WSN sítě, v které se pohybuje. Polohu sledovaného objektu můžeme určit tím, že zjistíme, ke kterému stacionárnímu prvku sítě se nachází náš sledovaný objekt nejblíže. Přesnost lokalizace pak závisí od množství jednotlivých stacionárních prvků rozmístěných v síti. Jelikož se sledovaný objekt v síti bude zřejmě pohybovat, bude zde docházet ke změnám v topologii sítě, přičemž bude také docházet k změnám v počtu prvků sítě, když se např. dostane sledovaný objekt mimo dosah sítě nebo naopak se dostane opět na dosah sítě.

2.3 Bezdrátové senzorové sítě pro monitoring

Tato skupina představuje WSN sítě, které jsou určené pro zabezpečení objektu nebo budovy (např. zabezpečovací systém využívající pohybové senzory nebo kouřová čidla atd.). Je složená ze stacionárních prvků, umístěných uvnitř sledovaného objektu, které mají za úkol sledovat jakoukoliv změnu od jejich klidového stavu a následně ji oznámit řídícímu prvku sítě. Nedochází zde k žádnému sbírání a s skladování dat, jak tomu bylo v předchozím případě, data jsou zde posílány k řídícímu prvku jen v případě, když senzor nebo čidlo zaznamená nějakou anomálii.

I v této síti se posílají po uplynutí určitém časové periody zprávy, které mají za úkol zjistit stav sítě. Jestliže by došlo k výpadku jakéhokoliv prvku nebo čidla v síti, nemuseli bychom tuto událost zaznamenat, přičemž to může být také známka narušení prostoru. Proto je nutné tuto událost zaznamenat a informovat řídící prvek sítě. Jelikož se zde neprovádí sběr dat, jsou zde menší nároky na paměť, ovšem vzrůstají požadavky na rychlost přenosu. Zprávu o narušení objektu potřebujeme přenést co možná nejrychleji k příjemci i za cenu větší spotřeby energie při přenosu, jelikož se předpokládá, že k těmto jevům bude docházet jen zřídka, tudíž nebudou mít za následek velkou spotřebu energie.

2.4 Bezdrátové senzorové sítě smíšené

Tento typ WSN sítí spojuje všechny dříve uvedené funkce. Může se např. jednat o sledování dopravní situace, kdy můžeme sbírat statistiky o počtu osobních aut nebo nákladních aut, jenž projedou určitou oblastí za hodinu, za den nebo za týden atd. Přičemž můžeme např. měřit jejich rychlost a v případě překročení maximální povolené rychlosti toto zaznamenat nebo kontaktovat policii.

2.5 Zigbee, 802.15.4

Pro komunikaci uvnitř WSN sítě se využívá ve velké míře technologie 802.15.4 zigbee. Tento standart je platný od roku 2004 a byl speciálně navržen pro bezdrátové personal area network WPAN, v kterých vyžadujeme v první řadě nízkou spotřebu energie. Jednotlivé koncové prvky PAN sítě jsou schopny komunikovat pouze na vzdálenost několika desítek metrů, tuto vzdálenost můžeme zvětšit použitím multihop směrovacího protokolu.

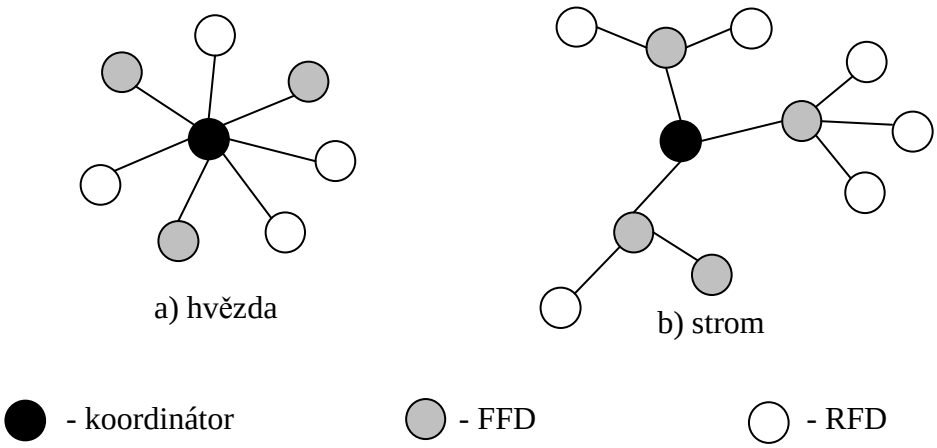
Zigbee je označován jako MAC (Media Access Control) layer protokol, tento protokol obsahuje mechanismus pro přístup ke komunikačnímu kanálu a stará se také o adresaci prvků v síti. K fyzickému médiu je přistupováno metodou CSMA/CA. Zařízení přistupující k fyzickému médiu metodou CSMA/CA nejprve poslouchá jestli na přenosovém médiu neprobíhá jiná komunikace, pokud komunikace neprobíhá, zařízení počká náhodnou dobu a po uplynutí této doby, pokud je médium i nadále neobsazené, začne vysílat. Pokud na přenosovém médiu komunikace probíhá, musí zařízení čekat na uvolnění média. Náhodný čekací interval před začátkem vysílání je zde použit z důvodu zabránění kolizím, ke kterým by mohlo dojít, kdyby na uvolnění přenosového média čekalo současně několik koncových zařízení.

Zigbee standart definuje tři různé prvky sítě koordinátor sítě C, zařízení s omezenou funkcí RFD (reduce function device) a zařízení s plnou funkcí FFD (full function device). RFD zařízení jsou určená pouze k měření určité veličiny nebo ke sledování určité události a jsou schopna odesílat zprávy směrem ke koordinátorovi, jedná se tedy o koncové zařízení sítě. FFD zařízení jsou schopna vykonávat funkci směrovače nebo mohou zastávat funkci koordinátora. S pomocí těchto tří zařízení, můžeme vytvořit tři různé topologie a to hvězdu, strom nebo kombinaci hvězdy se stromem. Zigbee standart se nevyznačuje závratnými

přenosovými rychlostmi, jelikož to není ani v sítích, pro které je určen potřeba. Nejedná se o konkurenci známého bluetooth, ale pouze o jakýsi jeho doplněk [7].

Pro zigbee byla definována tato přenosová pásma :

- Evropa 858 MHz (1 kanál) s přenosovou rychlostí 20 kbit/s
- Amerika a Austrálie 915 MHz (10 kanálů) s přenosovou rychlostí 40 kbit/s
- Globální 2,4 GHz (16 kanálů) s přenosovou rychlostí 250 kbit/s

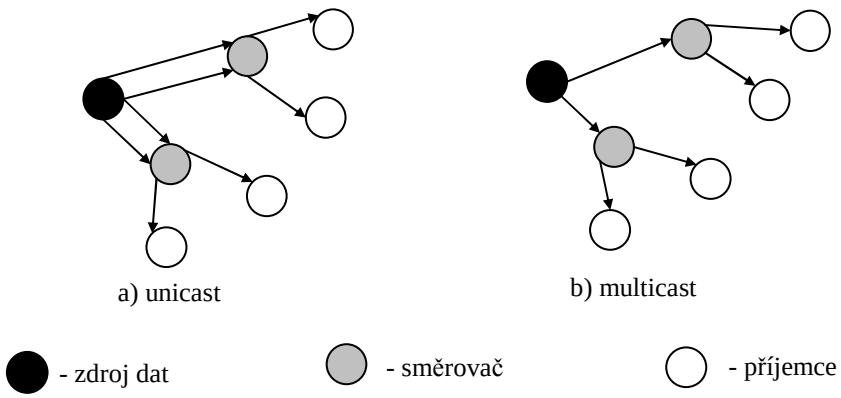


Obr.2.1: Topologie zigbee

3 Multicast

Jedná se o komunikaci typu jeden zdroj nebo několik zdrojů informace a více příjemců, přičemž zdroj vysílá data jenom jednou a o následné kopírování se starají jednotlivé uzly v síti, čímž dochází k výraznému zmenšení zátěže vysílacího uzlu, zvláště jedná-li se o velkou skupiny příjemců.

Dejme tomu, že máme 100 potencionálních příjemců v síti, kterým je nutno odeslat data o velikosti např. 1 MB, pokud bychom tyto data odesílali unicastově, musel by odesílatel odeslat postupně každému příjemci ona zmíněna data, čímž by došlo k odeslání 100MB dat od jednoho prvku sítě. Pokud ovšem bude komunikace probíhat formou multicastu, odesílatel odešle pouze 1MB dat. Data se vysílají každou větví sítě jenom jednou a to jen v případě, nachází-li se na ní potenciální příjemce. Cílová adresa je adresou multicast skupiny, jejíž členové jsou jednotlivý příjemci. U multicastu je složitější směrování a to zejména jedná-li se o poměrně rozsáhlou síť. Směrování musí zajistit to, aby se data vyslaná zdrojem dostala ke všem aktivním členům multicastové skupiny a pokud možno k nikomu jinému. K tomuto účelů byli vyvinuty multicastové směrovací protokoly např. PIM, DVMRP, MOSPF.



Obr. 3.1: Rozdíl mezi komunikací unicast a multicast

3.1 IP multicast

IP multicast umožňuje přenos dat od jednoho zdroje k několika příjemcům, data se samozřejmě od zdroje odesílají opět jen jednou, o kopírování dat a doručení se starají prvky sítě. Jelikož IP multicast využívá internet, příjemci se mohou nacházet kdekoliv na světě. Příjemci musí být členové multicastové skupiny, přičemž multicastová skupina je

identifikovatelná jedinou IP adresou. Členství v multicastové skupině je dynamické, kterýkoliv člen se může ze skupiny odhlásit nebo naopak do skupiny přihlásit, jeden uživatel však může být členem i několika multicastových skupin.

Multicastové skupiny můžeme rozdělit do dvou kategorií na permanentní a dočasné. Permanentní multicastové skupiny mají předem dané adresy, které jsou všem dobře známé a jsou používány pro specifické účely, tyto skupiny existují prakticky vždy a mohou mít i nulový počet uživatelů. Dočasné multicastové skupiny jsou sestavovány až v momentě, kdy jsou zapotřebí, pokud je počet jejich členů nulový zanikají. Rozdíl mezi unicastovým a multicastovým paketem poznáme pomocí jeho IP adresy. Pro multicastové pakety jsou vyhrazeny adresy z rozsahu D, tj. adresy, které leží v rozsahu 224.0.0.0 až 239.255.255.255. Některé z těchto adres mají zvláštní použití, jedná se o adresy permanentních multicastových skupin, tudíž nemohou být použity jako adresy dočasných multicastových skupin např. adresa 224.0.0.1 je adresa multicastové skupiny, kterou tvoří všechny prvky místní sítě, 224.0.0.2 jsou všechny směrovače v místní síti. Seznam těchto adres můžeme nalézt na http://www.iana.org/assignments/multicast-addresses, zkratka IANA znamená Internet Assigned Numbers Autority. IANA je organizace, která má za úkol dohlížet na celosvětové přidělování IP adres [1].

V dnešní době se IP multicast hojně používá při přenosu streamovaných dat např. video konference, internetová rádia, IP TV atd. Je to celkem logické, jelikož např. při IP TV máme již objemnější toky dat, které je třeba doručit relativně velkému počtu uživatelů.

Abychom mohli přenášet data IP multicastem, musí být sítí podporovány následující skutečnosti: přeposílaní paketů neboli forwarding, multicastové směrovací protokoly a IGMP protokol. IGMP protokol má za úkol odhlášení, přihlášení a udržování multicastové skupiny. Směrovací protokoly slouží k tomu, aby se multicastové pakety přeposílaly jen směry, kde se nacházejí členové multicastové skupiny a to pouze jednou např. protokol DVMRP, který využívá metodu reverse path forwarding (RPF).

4 Multicastové směrovací protokoly

Směrovací protokoly se používají k nalezení cesty v síti od odesílatele k příjemcům, přičemž při problémech v síti dochází k hledání alternativních cest. V dnešní době existuje velké množství jak unicastových, tak multicastových protokolů. Mezi nejpoužívanější patří RIP, OSPF, MOSPF, PIM, DVRMP.

4.1 PIM protokol

Zkratka PIM znamená protocol independent multicast, tento název byl zvolen, protože pro směrování nevyužívá žádný jiný protokol, tudíž není na jiném protokolu závislý. U PIM protokolu se ke směrování používají dva režimy hustý režim (dense mode) a řídký režim (sparse mode).

4.2 PIM hustý režim

Při použití hustého režimu (dense mode) se přepokládá, že multicastová skupina obsahuje velké množství členů, tudíž existuje velké množství příjemců, kteří tvoří většinu sítě. U tohoto režimu se data přeposílají všem v síti, přičemž uzel, který nemá o členství v multicastové skupině zájem, musí toto oznámit, čímž nám zůstanou ve skupině jen požadovaní uživatelé.

4.3 PIM řídký režim

Sparse mode nebo-li řídký režim předpokládá pravý opak a to, že v multicastové skupině se nachází jen hrstka účastníků v porovnání s velikostí sítě. Data se přeposílají až v okamžiku, požádá-li o ně některý z uzlů sítě. Zde ovšem narážíme na problém, který spočívá v tom, že máme data, ale nevíme komu je odeslat a naopak potencionální příjemci nevědí, kde se nachází odesílatel. Protokolem je proto vytvářen tzv. rendezvous point (bod setkání). Tento bod je každému v sítí předem dobře známý a slouží právě k setkání odesílatelů a příjemců multicastové skupiny.

4.4 DVMRP

DVMRP (distance vector multicast routing protokol) je jeden z prvních multicastových směrovacích protokolů, který byl použit v internetové síti. DVRMP využívá ke své činnosti IGMP protokol, pomocí něhož si aktualizuje informace o členství prvků v multicastových skupinách. Směrovače v síti si uchovávají tabulku s adresou zdroje a adresou multicastové skupiny (S,G). DVMRP si kvůli směrování udržuje i unicastovou směrovací tabulku [2].

Ke směrování využívá metody reverse path algritmus. O přeposílání neboli anglicky forwarding multicastových paketů v síti se starají směrovače. Směrovač musí přeposlat paket směry, kde se nacházejí potenciální příjemci nebo-li členové multicastové skupiny. Nesmí dojít k odeslání ve směru, odkud paket přišel nebo směrem, kde již byl paket jednou poslán, pokud by k tomuto docházelo hrozilo by neustále přeposílání paketů z jednoho směrovače na druhý až do vynulování Time To Live, čímž by docházelo ke zbytečnému zatěžování sítě, přičemž by členům multicastové skupiny byli zasílány duplikované pakety.

Určitě si říkáte, že řešení je úplně jednoduché, jednoduše pakety se nebudou přeposílat směrem odkud je směrovač obdržel. Toto je určitě z jisté části řešení, ale ne úplně dostatečné. Představme si jednoduchou situaci, kdy máme tři směrovače X, Y, Z spojené způsobem každý s každým. Když nám ze směrovače X přijde multicastový paket, přičemž členové multicastové skupiny se nacházejí na rozhraních obou směrovačů Y, Z, obdrží tedy tuto zprávu i směrovače Y,Z, přičemž ji přepošlou správným směrem, tedy na místo kde se nachází uživatel multicastové skupiny, ale jelikož jsou vzájemně spojeni a směrovač Y ví, že se člen multicastové skupiny nachází i na rozhraní směrovače Z a naopak, přepošlou si tedy paket také navzájem. K přeposlání dojde pouze jen dvakrát, ale naše síť byla velice jednoduchá, představte si rozsáhlou síť, v takovém případě by duplikovaných paketů bylo určitě daleko víc.

K předcházení duplikaci paketů se používá metody Revrse Path Forwarding (RPF). Tato metoda spočívá v tom, že směrovač musí znát cílovou adresu multicastové skupiny a také adresu zdroje. Pokud směrovači přijde multicastový paket, zjistí podle adresy zdroje, jestli by stejným rozhraním, na které mu paket přišel poslal unicastový paket opačným směrem tedy směrem ke zdroji, jestliže tomu tak je, směrovač se postará o přeposlání. Pokud by však tomu tak nebylo, směrovač paket zahodí. Vezměme si opět náš příklad se třemi směrovači X,Y,Z. X pošle multicastový paket k Y a Z. Y a Z zjistí, že stejným směrem, kterým paket přišel by odeslali i oni paket směrem ke zdroji, a proto přepošlou paket dále,

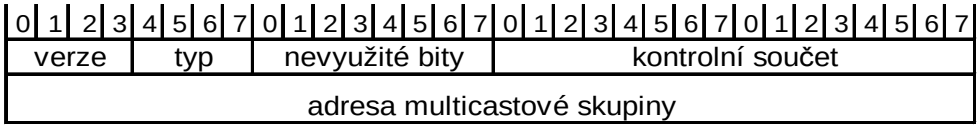
čímž sice opět dojde k odeslání paketu od Y směrem k Z a naopak, ovšem tyto pakety již směrovače nepřepošlou dále, jelikož zdroj se nachází na jiném rozhraní a dojde tak k jejich zahození. Jelikož potřebujeme také zjišťovat, jakým směrem bychom odeslali paket unicastově směrem ke zdroji, směrovače si uchovávají také unicastovou směrovací tabulku.

5 IGMP protokol

IGMP (Internet group managment protocol) protokol má za úkol výměnu informací o členství uživatelů v jednotlivých multicastových skupinách. Abychom mohli sestavit multicastovou skupinu, musíme nejprve najít její členy. K výměně informací dochází v pravidelných časových intervalech, tyto informace jsou vysílány členy multicastových skupin.

5.1 IGMPv1

IGMPv1 rozlišuje dva typy zpráv host membership report (zpráva o členství prvku v multicastové skupině), host membership query (dotaz na členství v multicastové skupině). Host membership report zpráva je odesílána jako odpověď na zprávu membership querry a zároveň informuje o nově nabytém členství prvku v multicastové skupině. Zprávy host membership query se posílají v pravidelných časových intervalech (např. jednou za minutu) sítí a slouží k zjištění platnosti členství klientů v multicastové skupině. Pokud nedojde odpověď na tři po sobě jdoucí zprávy, uzel je vyloučen z multicastové skupiny a zprávy membership query již mu nejsou zasílány.



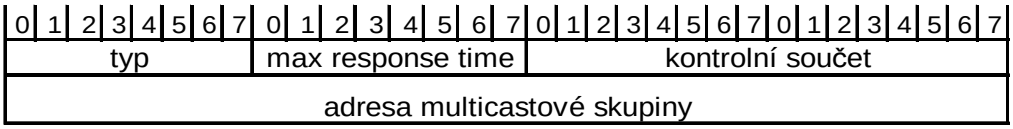
Obr. 5.1 Formát IGMPv1 zprávy

Verze – označuje verzi IGMP protokolu, Typ – označuje typ posílané zprávy membership query nebo membership report, Kontrolní součet – slouží pro příjemce k určení, nedošlo-li při přenosu k chybám, Adresa multicastove skupiny – je nastavená jen v případě, posílá-li se zpráva membership report v ostatním případech je nastavena na 0.0.0.0 [1].

5.2 IGMPv2

IGMPv2 obsahuje na rozdíl od IGMPv1 další typ zprávy navíc s názvem leave group. Leavse group se vysílá v případě chystá-li se uzel opustit multicastovou skupinu. Přijme-li řídící prvek zprávu leave group, odešle obratem odpověď ve formě zprávy Group Specific

Query , aby zjistil jestli se někdo nachází ve zbylé multicastové skupině, pokud tomu tak není, dochází ke zrušení této multicastové skupiny. Leave group zpráva má za následek snížení zpoždění při odhlašování uzlu z multicastové skupiny. IGMPv2 je kompatibilní s IGMPv1.



Obr. 5.2: Formát IGMPv2 zprávy

Typ – Pomocí hodnot nastavených v tomto poli rozeznáváme nejen typ zprávy, ale také o jakou verzi IGMP se jedná. Rozeznáváme tedy typy membership querry – kód 11 (existují tady vlastně dva typy membership query a to General Query a Group specific Query, obě zprávy mají stejný kód, rozeznávají se mezi sebou dle hodnoty v poli adresa), membership report pro IGMPv1 – kód 12, membership report pro IGMPv2 – kód 16, Leave Group message – kód 17 , Max response time – tato položka bývá využita při odesílání zprávy query a udává maximálni dobu (bývá v rámci několika sekund), po kterou se čeká na odpověď na tuto zprávy, Kontrolní součet – slouží k zjištění, zda nedošlo při přenosu k chybám, Adresa multicastové skupiny – je nastavena jen v případě, pokud se přenášejí zprávy Leave Group, Membership Report a Group Specific Query, u ostatních typů správ je nastavena na 0.0.0.0 [4].

5.3 IGMPv3

IGMPv3 se snaží řešit problém člena multicastové skupiny, který nechce přijímat zprávy od všech zdrojů, ale pouze od některých. Došlo zde k výrazné změně formátů zpráv. Jednotlivé zprávy nedisponují konstantní délkou, jak tomu bylo v předchozích dvou případech. Pole max response time se mění na Max Response code, přičemž udává pořád dobu, po kterou se čeká na odpověď, tuto dobu však udává jen v případě, pokud je hodnota menší než 128 (tzn. MSB = 0). Typy zpráv v IGMPv3: IGMPv3 membership report kód 22, IGMPv2 membership report kód 16, IGMPv1 membership report kód 12, IGMPv2 leave group kód 17, membership query kód 11 [5].

6 Ad-Hoc sítě

Slovní spojení Ad-hoc znamená v překladu pro specifický účel, jedná se tedy o vytvoření spojení dle potřeby pro specifický účel. Ad-hoc spojení bývá vytvořeno až v momentě nastane-li potřeba komunikace v síti, po přenesení potřebných informací se toto spojení zruší, bývá tedy sestaveno většinou za běhu a obvykle jen dočasně. Tento typ spojení je často používán v bezdrátových sítích. Ad-hoc síť může být vytvořena skupinou koncových stanic, které se nacházejí ve své blízkosti a jsou schopny komunikovat každý s každým. Jelikož bezdrátové sítě mají jen omezený dosah, ve většině případů stanice nejsou schopny komunikovat každý s každým, proto Ad-hoc sítě mohou být koncipované také jako „multihopové“ (na posílání paketů se podílejí také mezilehlé stanice).

Z výše uvedeného můžeme usoudit, že Ad-hoc síť nepotřebuje ke své činnosti žádný prostřední prvek neboli přístupový bod (Access point), čímž se její výstavba stává jednoduchá a levná, přičemž rozšíření sítě nebo přidání nových zařízení také není žádný velký problém. Díky tomu, že spojení je vytvořeno jen po určitou potřebnou dobu, dochází tím k výraznému šetření energie, tento způsob komunikace je tímto předurčen pro systémy, které jsou napájeny z bateriových zdrojů a požadujeme u nich velkou životnost, jednalo by se zřejmě zejména o senzorové sítě, ve kterých je potřeba informaci přenést jen v případě zaznamená-li čidlo určitou událost.

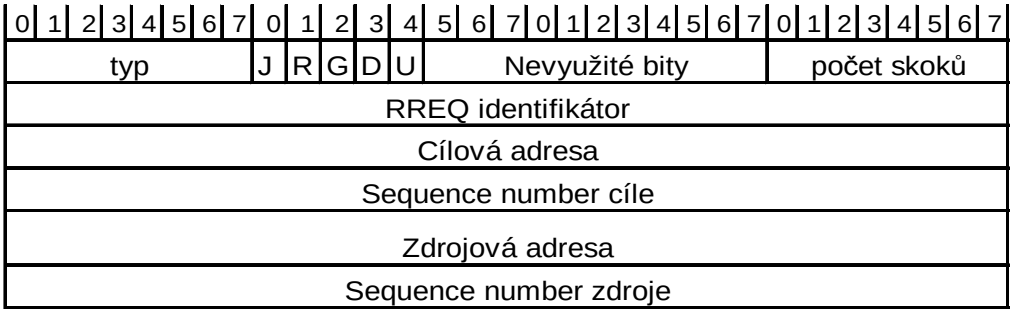
6.1 Ad-hoc směrovací protokoly

Směrovací protokoly v Ad-hoc sítích bychom mohli rozdělit do tří skupin reaktivní, proaktivní a hybridní. Reaktivní směrovací protokoly se vyznačují tím, že si neudržují směrovací tabulky, cestu v síti hledají nebo vypočítávají až v momentě, kdy je potřeba. U proaktivních směrovacích protokolů dochází nejprve k sestavení směrovacích tabulek, které jsou udržovány za účelem pozdějšího využití. K aktualizaci údajů v tabulkách dochází v periodických intervalech nebo pokud nastane změna v síti (výpadek linky, přidání nových prvků do sítě atd.). U hybridních směrovací protokolů dochází ke kombinací vlastností reaktivních a proaktivních směrovacích protokolů.

6.2 AODV

Zkratka AODV znamená Ad Hoc On Demand Distastance Vector směrovací protokol. Jedná se o unicastový směrovací protokol používaný v bezdrátových Ad-hoc sítích. AODV se řadí do skupiny tzv. reaktivní protokolů, což znamená, že spojení se ustanovuje jen v případě potřeby komunikace (On Demand = na vyžádání). AODV podporuje unicast, pro multicast se používá směrovací protokol MAODV.

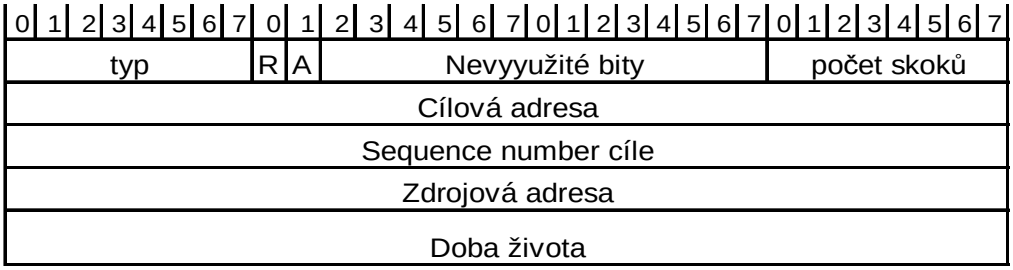
Cesta v síti je hledána na základě dotazů a následných odpovědí, využívají se tři druhy zpráv RouteRequest, RouteResponse, RouteError. Tyto zprávy jsou odesílány UDP protokolem. Pokud potřebuje např. uzel C komunikovat v síti s uzlem L, vyšle uzel C všesměrově (broadcast) žádost o spojení RouteRequest (RREQ), ostatní uzly tuto žádost přeposílají dále, přičemž inkrementují hodnotu hop count v RREQ a uloží si však informaci z kterého uzlu žádost přišla. Tímto nám vznikne prakticky řetězová reakce, kdy každý uzel, který žádost RouteRequest obdrží, pošle tuto žádost ke všem svým dostupným uzlům. Pokud žádost RouteRequest obdrží uzel v jehož dosahu je uzel L nebo samotný uzel L, vyšle zprávu RouteReply (RREP) zpět k uzlu, od kterého RREQ obdržel. Přičemž všechny uzly, kterými prochází RREP zpět k uzlu C si aktualizují své směrovací tabulky (routing tables), aby je mohly použít pro sestavení reverzní cesty. Uzel C takto může získat více cest k požadovanému uzlu L, vybere si však tu, které prochází přes nejmenší počet uzlů (má nejmenší počet skoků).



Obr. 6.1: Formát zprávy RREQ

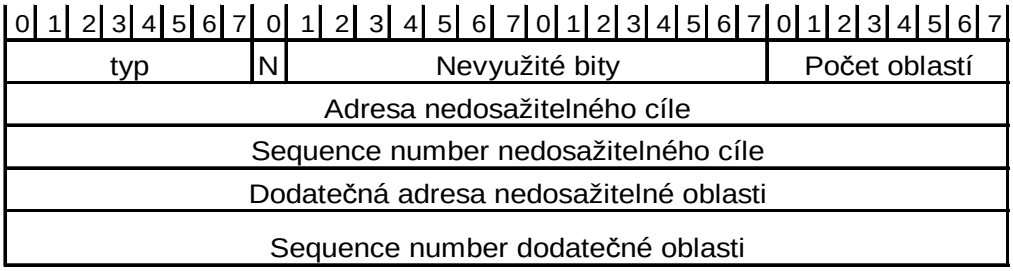
Typ – určuje typ zprávy (1 = RREQ), J – používá se při multicastu, pokud je bit nastaven, jedná se o zprávu JOIN, R – používá se při multicastu, pokud je bit nastaven, jedná se o zprávu REPAIR, G – po doručení RREQ k uzlu, který má ve své směrovací tabulce záznam o cílovém uzlu dochází k zahození RREQ a k odeslání RREP směrem ke zdroji. Pokud k tomuto dojde a bit G je nastaven, musí být dodatečně poslána zpráva RREP

k cílovému uzlu, D - tento bit nám říká, že pouze cílový uzel může odpovědět, U – udává neznámé sequence number, Nevyužité bity – tyto bity se při přijmu ignorují, Počet skoků – hodnota tohoto pole je inkrementována při každém přeposlání zprávy, RREQ identifikátor – identifikační číslo, které ve spojitosti s adresou zdroje tvoří jedinečný identifikační údaj každé RREQ, Cílová adresa – Adresa cílového uzlu, Sequence number cíle – poslední sequence number cílového uzlu přijaté v minulosti, pokud tento údaj nemáme, provede se nastavení bitu U, Zdrojová adresa – adresa odesílatelem, Sequence number – současné sequence number cíle [3].



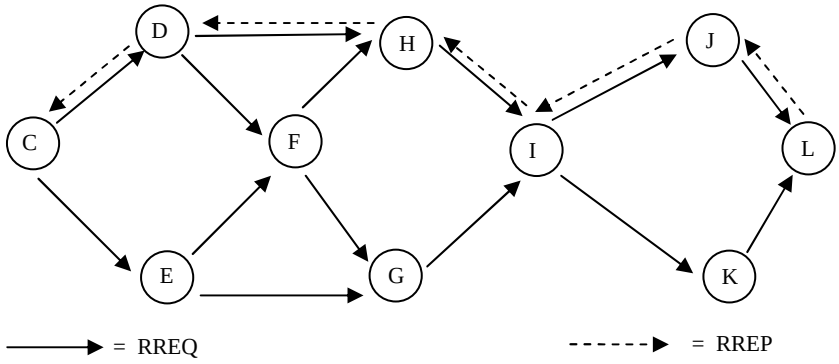
Obr. 6.2: Formát zprávy RREP

Typ – určuje typ zprávy (2=RREP), R - používá se při multicastu, pokud je bit nastaven, jedná se o zprávu REPAIR, A – tento bit je nastaven, pokud požadujeme potvrzení ACK o přijetí zprávy RREP, Počet skoků – počet skoků ke zdrojovému uzlu, Cílová adresa – adresa prvku, který RREP posílá, Sequence number cíle – sequence number cílového uzlu, Zdrojová adresa – adresa odesílatele RREQ, Doba života – udává dobu v milisekundách, po jejímž vypršení se zpráva zahodí.[3]



Obr. 6.3: Formát zprávy RERR

Typ – určuje typ zprávy (3=RERR), N – značí, že uzly, které tuto zprávu obdrží nebudou mazat své záznamy ve směrovací tabulkách, Počet oblastí – počet nedosažitelných uzlů, Adresa nedosažitelného cíle – Adresa oblasti, jenž je nedostupná kvůli problémům v síti, Sequence number nedosažitelného cíle – Sequence number nedosažitelného cíle, které má uzel, jenž RERR odesílá uložen ve své směrovací tabulce. [3]



Obr. 6.4: Posílání zprávy RREQ a RREP

Po přijmutí RREP se mohou začít okamžitě odesílat data, přijme-li však odesílatel další RREP pokud již zahájil odesílání dat a pokud tato cesta obsahuje menší počet skoků, může začít využívat tuto novou výhodnější cestu. Z obrázku 6.4 můžeme vidět hlavní nevýhodu této metody, jedná se o zaměstnávání prakticky všech uzlů v síti zprávu RREQ. V případě malé sítě se nejedná o tak závažný problém, ale v případě rozsáhlejší sítě, je toto zbytečné, zvlášť když hledaný uzel leží někde poblíž. Jedno z řešení tohoto problému by bylo nastavovat maximální počet skoků v zprávě RREQ nejprve na malé hodnoty, které bychom, pokud by nepřišla odpověď, postupně zvětšovali.

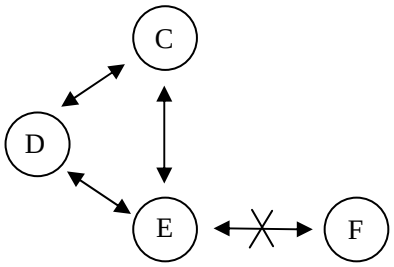
Cesta je považována za aktivní do té doby, dokud jí procházejí od odesílatele k příjemci data. Jakmile se přestanou posílat pakety sítí, spojení je přerušeno. K přerušení spojení může dojít také během posílání datových paketů, pokud k tomuto dojde, uzel, který rozpad spojení zaznamená, odešle zprávu RouteError (RERR) zpět k odesílateli. Jakýkoli uzel, který přijme zprávu RouteError, vymaže ze své směrovací tabulky všechny cesty obsahující nedostupný uzel. Pokud chce odesílatel znovu začít komunikovat s příjemcem, musí znovu poslat zprávu RouteRequest a čekat na odpověď. Každá RouteRequest s sebou nese SequenceNumber, toto SequenceNumber nám zajistí, aby uzly znovu nepřeposílali RouteRequest, kterou již jednou obdrželi. Uzly aktualizují svoje informace o cestě v sítí jen v případě, že RouteRequest obsahuje vyšší SequenceNumber než to, které mají uložené v RoutingTable. RoutingTable slouží ke skladování informací o aktivních cestách, ke

krátkodobému uložení informací při přijmu zprávy RREQ pro pozdější sestavení zpětné cesty směrem k odesílateli.

RoutingTable obsahuje následující informace:

- adresu cílového uzlu
- adresu sousedního uzlu
- SequenceNumber cílového uzlu
- metriku cesty nebo-li počet skoků (hopů) k příjemci
- doba života

Informace jsou ve směrovací tabulce uloženy až do vypršení doby života. Adresa sousedního uzlu slouží k určení směru, kterým mají být data k příjemci poslána. Pokud uzel zná cestu k příjemci, ovšem její SequenceNumber je nižší než SequenceNumber v RouteRequest, nemůže odpovědět na RREQ musí přeposlat RouteRequest dál a tím si aktualizovat informace o daném cíli. Sequence number nám zajistí také to, že nedojde k zacyklení při zjišťování cesty. Uvažujme např. síť dle Obr. 6.5 s uzly C,D,E,F.



Obr. 6.5: Vznik zacyklení

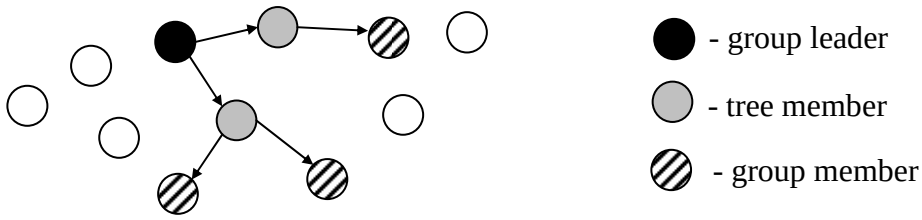
Dejme tomu, že došlo k přerušení spojení mezi uzly E, F , jak je naznačeno na Obr. 6.5. Uzel D má stále uvedeno ve směrovací tabulce, že k F vede cesta přes E o vzdálenosti 2 skoků. E však ví, že cesta k F je přerušena. Jakmile bude chtít C aktualizovat údaje o spojení s F, ve své směrovací tabulce zjistí, že k F se dostane přes D o vzdálenosti 3 skoků, E vzápětí zjistí, že má cestu k F přes C o vzdálenosti 4 skoky, přičemž A zjistí že k F vede cesta přes E o vzdálenosti 5 skoků a takto by to pokračovalo donekonečna.

Díky SequenceNumber však C zjistí, že cesta k F přes D je už zastaralá. B tedy bude hledat cestu k F pomocí RouteRequest. Pokud po odeslání RouteRequest uzel neobdrží odpověď, musí počkat dvojnásobnou dobu než je timeout předchozí zprávy, než může odeslat další zprávu. Každý uzel může pomocí tzv. Hello Message zjistit uzly, které leží v jeho těsné blízkosti. Hello Message je prakticky RouteRequest s TimeToLive nastaveným na 1, tím se zaručí, že se RouteRequest dostane jen na vzdálenost 1 skok od odesílatele nebo-li nebude

nikdy přeposlána, přičemž dojde samozřejmě k aktualizování směrovacích tabulek sousedních uzlů. Tyto HelloMeassage mají také ještě jinou funkci a to zjištění stavu linky.

6.3 MAODV

Zkratka MAODV znamená Multicast Ad hoc On Demand Distance Vector, jedná se o multicastový směrovací protokol odvozen od protokolu AODV. MAODV pracuje na stejném principu jako AODV s malými obměnami. Abychom mohli komunikovat s více uzly najednou, musíme tyto uzly spojit do tzv. stromové struktury. Tato stromová struktura se skládá nejen z uzlů, kteří jsou členové multicastové skupiny, ale také z mezilehlých uzlů, kteří tvoří pouze prostředníka v předávání paketů, bez jejich pomoci by totiž nebylo možné pakety doručit k ostatním členům skupiny. Každá stromová struktura se skládá ze tří typů členů a to člen multicastové skupiny (multicast group member), člen stromové struktury (multicast tree member) a vedoucí skupiny (group leader), přičemž multicast tree member nemusí být vždy ve stromové struktuře zastoupen. Group leader se stává automaticky uzel, který dal podnět k vytvoření skupiny.



Obr. 6.6 Multicastový strom

Multicastová skupina je jednoznačně určena adresou skupiny (group address) a sequence number, které udává obdobným způsobem jak v AODV novost cesty ve skupině. Group leader je zodpovědný za udržování novosti spojení multicastove skupiny. K udržování novosti spojů v multicastove skupiny využívá group leader tzv. Group Hello (GRPH) zpráv. Tyto zprávy jsou periodicky posílány v síti, přičemž s každou novou GRPH vždy dochází ke zvyšování hodnoty o 1 v položce group sequence number. Směrovací tabulka uzlu v multicastové skupině obsahuje více informací, než tomu bylo u protokolu AODV.
Směrovací tabulka MAODV obsahuje následující informace:

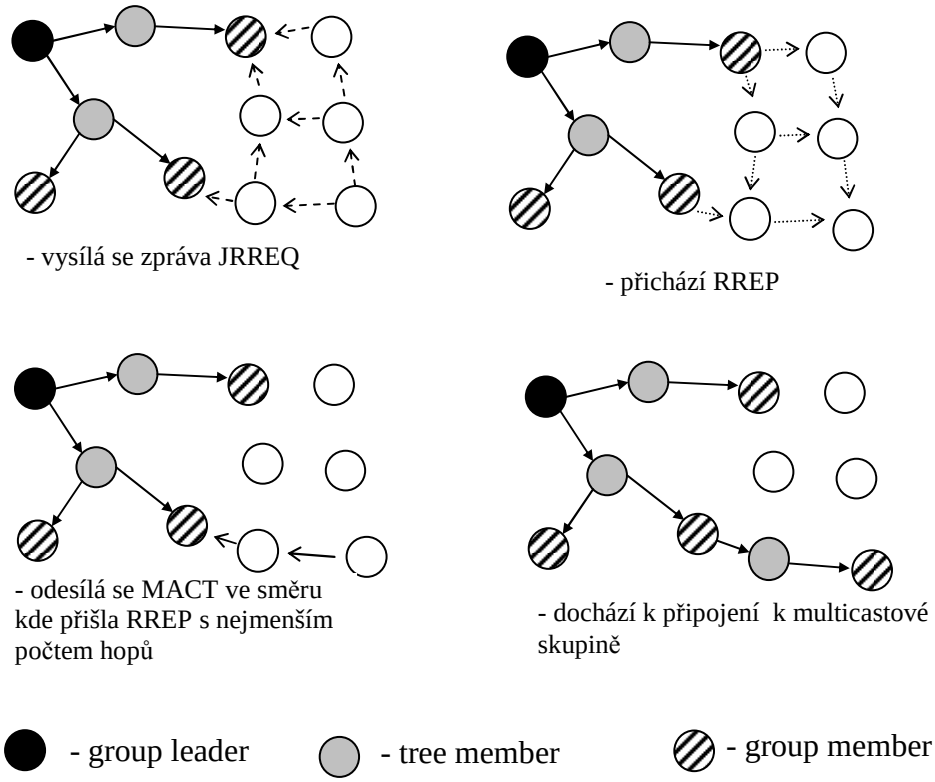
- adresu multicastové skupiny
- adresu vedoucího multicastove skupiny
- počet hopů k vedoucímu multicastove skupiny
- Sequence number

- Time to live

Uzel, který chce komunikovat se skupinou jiných uzlů, vyšle zprávu Route Request (RREQ) s adresami všech uzlů budoucí multicastové skupiny, tato zpráva se šíří všesměrově (broadcast), přičemž odpověď Route Reply (RREP) přichází od všech členů multicastové skupiny. V MOADV směrovacím protokolu rozeznáváme dva typy RREQ. První typ RREQ vysílá uzel, který chce vytvořit neexistující multicastovou skupinu a stává se automaticky jejím group leaderem. [4]

Druhý typ se nazývá join route request (JRREQ), která se vysílá v případě, že se uzel chce připojit k existující multicastove skupině. Pouze člen stromové struktury v multicastové skupině může odpovědět na zprávu JRREQ. Následná odpověď RREP obsahuje destination sequence number, group leader address, počet skoků ke group leaderu, počet skoků ke stromové struktuře. Pokud by se jednalo o uzel, který je zároveň členem stromové struktury, položka počet skoků ke stromové struktuře se nastaví na 0. Zpráva route reply u MAODV, neustanovuje spojení, jak tomu bylo u AODV, pouze poskytuje jednu z možných cest k připojení do stromové struktury. Pro přidání nové větve ke stromové struktuře složí zpráva MACT (multicast route activation).

Jestliže tedy máme uzel, který se chce připojit k existující stromové struktuře, tento uzel nejprve musí vyslat zprávu JRREQ, přičemž dostane odpovědi RREP od více prvků stromové struktury, jelikož jednotlivý členové stromové struktury se vzájemně neinformují o příchozích JRREQ. Uzel si pak vybere z jednotlivých RREP tu, která je nejvýhodnější a odešle aktivační zprávu MACT, čímž je vybraná cesta aktivována a připojena ke stromové struktuře.

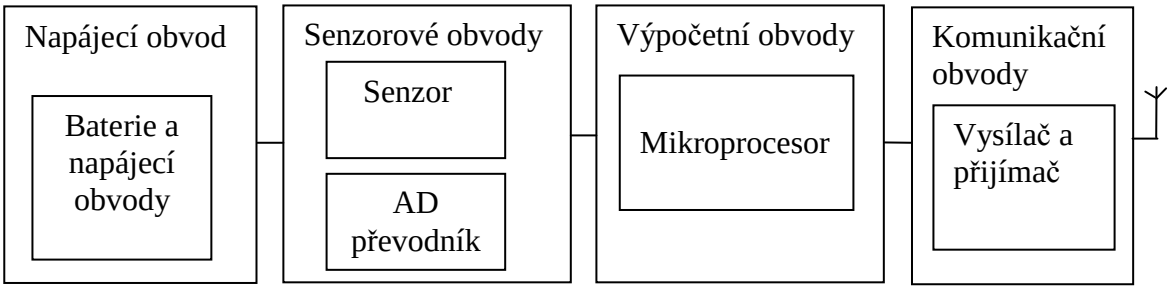


Obr 6.7 Připojování se k existující multicastové skupině

Zpráva MACT tedy obsahuje adresu zdroje, sequence number zdroje a adresu multicastove skupiny. Uzel může kdykoliv multicastovou strukturu opustit. Pokud by se jednalo o koncový uzel, dojde jednoduše k jeho vypuštění ze stromové struktury. Pokud se však jedná o uzel, který zprostředkovává komunikaci i jiným prvků multicastové skupiny, dojde k vypuštění tohoto uzlu z multicastové skupiny, zůstane však ve stromové struktuře, aby plnil funkci prostředníka při komunikaci (tree member). K odstranění uzlu z multicastove skupiny slouží zpráva prune MACT (P_MACT) .

7 Architektura uzlu bezdrátové senzorové sítě

Uzel v bezdrátové senzorové síti je složen ze čtyř základních bloků napájecí obvody, senzorové obvody, výpočetní obvody a komunikační obvody. Napájecí obvody zajišťují napájení všech ostatních bloků, přičemž musí být schopné zajistit skokovou změnu napájecího proudu při změně z režimu spánku (desítky mikroampér) do aktivního režimu (desítky miliampér). Senzorové obvody sestávají z několika senzorů a z obvodů, které slouží pro ovládání těchto senzorů. Obsahují digitální nebo analogový výstup, který zprostředkovává čtení naměřených hodnot. Výpočetní obvody se skládají z mikroprocesoru, pamětí, časovačů, vstupních a výstupních obvodů, ADC převodníku a ostatních periferií, přičemž výběr mikroprocesoru výrazně ovlivňuje spotřebu energie uzlu. Komunikační obvody umožňují komunikaci uzlu s ostatními uzly a jsou tvořeny bezdrátovým přijímačem, vysílačem a anténou.



Obr. 7.1: Blokové schéma uzlu v bezdrátové senzorové síti [6]

7.1 Spotřeba energie prvku v bezdrátové senzorové síti

Spotřeba energie v každém prvku bezdrátové senzorové sítě sestává ze součtu spotřeb jednotlivých úkonů, které jsou prováděny uvnitř uzlu. Tyto úkony bychom mohli rozdělit do třech bodů sběr dat, komunikace a výpočetní úkony. Jelikož energii spotřebovává v největší míře hardware, nejvíce úspory se dosahuje při samotném návrhu.

1) Sběr dat:

   Při sběru dat musí nejprve dojít k aktivaci senzorových obvodů určených pro snímaní okolního prostředí a pro sběr dat z okolního prostředí. Spotřeba energie při sběru dat závisí na snímané veličině a na způsobu jejího snímaní, rozdílné senzory spotřebovávají pro snímání a sběr dat různé množství energie.

2) Komunikace:

   Spotřeba energie uzlu při komunikaci v bezdrátové senzorové sítí sestává ze dvou částí a to spotřeba při odesílání paketů (tx power) a spotřeba při přijímání paketů (rx power), obě hodnoty závisí zejména od maximálního dosahu přijímací a vysílací části..

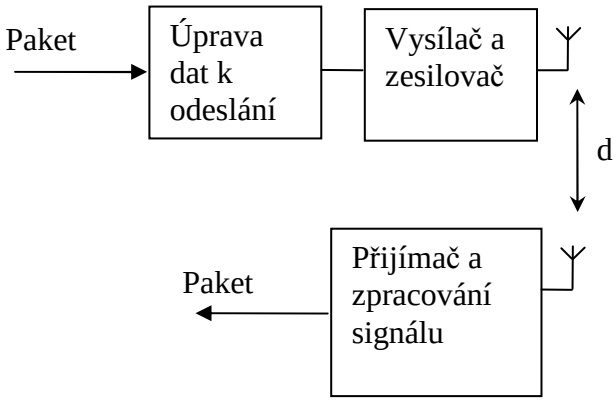
3) Výpočetní úkony:

   Výpočetní úkony zajišťuje mikroprocesor, tyto úkony jsou potřeba ke každé činnosti, kterou uzel provádí. Zejména jsou důležité při sběru dat, kdy dochází k dodatečným úpravám sesbíraných dat. Spotřeba energie při výpočetních úkonech je na rozdíl od komunikace a sběru dat poměrně malá.

7.2 Spotřeba energie komunikačních obvodů

Komunikační obvody sestávají z bezdrátového přijímače a vysílače, zajišťují komunikaci s ostatními prvky sítě. Bezdrátová komunikace může probíhat na základě radiových vln nebo se může jednat o komunikaci založenou na optickém přenosu. V případě použití bezdrátového optického přijímače a vysílače musíme zajistit vzájemnou viditelnost mezi komunikujícími uzly, což není vždy možném. Optická komunikace na rozdíl od rádiové je však energeticky méně náročná.

Spotřeba energie v případě rádiového přenosu sestává ze spotřeb obvodů, které zajišťují rádiový přenos a elektronických obvodů, které slouží pro přípravu dat k odeslání. Blokové schéma komunikačních obvodů je zobrazeno na obrázku 7.2.



Obe. 7.2: Blokové schéma komunikačních obvodů [9]

Spotřeba energie vysílací části závisí na velikosti paketu, který se odesílá, na použité modulaci a na vzdálenosti, na kterou komunikace probíhá a na vysílacím výkonu zesilovače. Pro výpočet celkové spotřeby energie při vysílacím procesu platí vzorec 7.1. [9]

$$Etx = Em \cdot n + Ez \cdot n \cdot d^2 \tag{7.1}$$

Kde *Etx* je spotřebovaná energie v joulech při vyslání paketu o *n* bitech na vzdálenost *d* metrů, přičemž *Em* je energie, která se spotřebuje při modulaci jednoho bitu a *Ez* je energie spotřebovaná zesilovačem.

Příjem je energeticky méně náročný, spotřeba energie závisí na počtu přijímaných bitů a na spotřebě energie obvodů, které slouží pro zpracování přijímaného signálu. Pro výpočet energie spotřebované při přijímacím procesu platí vzorec 7.2.[9]

$$Erx = Ed \cdot n \tag{7.2}$$

Kde *Erx* je spotřebovaná energie v joulech pro příjem paketu o *n* bitech a *Ed* je spotřeba energie obvodů pro příjem a demodulaci signálu na jeden bit.

Jelikož energetické nároky elektronických obvodů uvnitř uzlu mohou ovlivnit pouze výrobci, můžeme spotřebu v tomto směru ovlivnit správným výběrem zařízení, jedná se zejména o spotřebu mikroprocesoru, spotřebu uzlu v aktivním režimu, spotřebu uzlu v režimu spánku (v kterém tráví většinu času), spotřebu při vysílání a přijímání (použitá komunikační technologie), spotřebu při aktivních senzorových obvodech atd. Spotřebu energie elektronických obvodů při komunikaci však můžeme ovlivnit omezením maximálního možné dosahu (snížením vysílacího výkonu), výběrem směrovacího protokolu, způsobem přenosu (multicast, unicast, broadcast) a výběrem protokolu pro přenos dat.

8 Network Simulator 2

NS2 (network simulator 2) je simulační nástroj, který umožňuje sestavit model sítě a následně ověřit chování tohoto modelu. Jedná se o objektově orientovaný nástroj, který obsahuje celou řadu síťových prostředků pro simulaci, jako např. TCP, směrovací protokoly, unicast, multicast, bezdrátové sítě. Dokonce zde můžeme implementovat i vlastní protokol a následně odsimulovat jeho chování. NS2 vychází z jazyka C++, ale simulační skripty jsou psány v jazyku TCL. Jednou z velkých výhod tohoto simulačního nástroje, je jeho volné stažení na stránkách http://www.isi.edu/nsnam/ns , kde také najdeme návody na instalaci a následné používání. Program je navržen pro používání na operačních systémech běžících na unixovém jádře, jsme však schopni při použití virtuálních nástrojů (např. Cygwin u Windows, VMWare player u linuxu) pracovat s NS2 i na jiných platformách než unixových (návody na instalaci jsou k dispozice na výše zmíněných stránkách) [8].

Základní operace, které umožňuje NS2 provádět jsou budování modelu sítě, stanovení cesty v modelu sítě, budování spojení, generování provozu, modelování poruch a monitorování provozu. Všechny tyto operace jsou zapisovány jako příkazy TCL do textového souboru, můžeme je psát v kterémkoliv textovém editoru. Znázornění výsledků probíhá nezávisle na simulační nástroji NS.

Grafické znázornění výsledků simulace umožňuje nástroj NAM (network animator). Pro zobrazení časově závislých průběhů simulace slouží nástroj xgraph. NAM je napsán v jazyce TCL/TK, jedná se o grafický nástroj, s jehož pomocí si můžeme prohlédnout kdy a co bylo v síti odesláno, přijato, zahozeno, jaká byla reakce na výpadek linky v síti, můžeme sledovat námi vybraný paket při cestě sítí atd. Abychom mohli použít nástroj NAM pro grafické znázornění výsledků simulace, musíme nejprve vytvořit soubor tracefile. Tento soubor bývá obvykle vytvořen ns simulátorem a obsahuje informace o topologii sítě, prvcích v síti, o vysílání a přijímání paketů, o frontách atd. Jakmile máme vytvořený soubor tracefile, můžeme zobrazit výsledky simulace v nástroji NAM. Při spuštění NAM tedy dochází k přečtení souboru tracefile, následně k vytvoření topologie sítě, otevření okna NAM console a otevření okna NAM, ve kterém dojde k vykreslení vytvořené topologie. Zobrazení výsledků simulace můžeme pozorovat po kliknutí na tlačítko PLAY v okně NAM, můžeme zde také měnit rychlost zobrazení výsledků atd. Okno NAM console slouží k tomu, abychom mohli spustit několik animací pod jednou NAM instancí, po kliknutí na FILE -> NEW můžeme otevřít jakýkoliv existující tracefile.

8.1 Konfigurace modelu sítě v NS2

Při vytváření TCL skriptu pro NS2 musíme v první řadě vytvořit objekt, který bude reprezentovat simulátor, toto se provádí příkazem `set ns [new Simulator]`, kde proměnná ns odkazuje na instanci tohoto objektu. Dále vytvoříme soubory pro zapisování výsledků naší simulace `set tracef [open vysledky_simulace.tr w]`, tímto jsme vytvořili soubor výsledky_simulace.tr, do kterého budou zapisovány výsledky naší simulace v textové podobě a pomocí `set namtr [open vysledky_simulace.nam w]` vytvoříme soubor, který slouží pro grafické znázornění výsledků simulace. Typ souboru, který takto vytváříme, závisí na způsobu, jakým chceme výsledky simulace interpretovat. Samozřejmě pokud požadujeme více interpretací výsledků naší simulaci, vytváříme takto hned několik souborů. Simulátoru ještě musíme říct, aby výsledky simulace zapisoval do námi vytvořeného souboru, což provedeme pomocí příkazu `$ns_ trace-all $tracef` respektive `$ns namtrace-all $namtr` pro soubor nam.

Dále pak vytvoříme proceduru, která bude volána při skončení simulace, tuto proceduru můžeme pojmenovat např. konec a její podoba bude následující.

```
proc konec {} {
    global ns tracefd
    global ns namtr
    $ns flush-trace
    close $tracef
    close $namtr
    exec nam vysledky_simulace.nam &
    exit 0
}
$ns run
```

Touto procedurou zavřeme soubory, do kterých se zapisovaly výsledky simulace a spustíme pomocí nástroje nam soubor vysledky_simulace.nam. Tato procedura bude volána příkazem `$ns at (čas ukončení simulace) konec` a budeme ji volat až po skončení všech událostí. Za procedurou konec je ještě příkaz, který slouží pro spuštění simulace. Tímto jsme vytvořili jednoduchý skript v jazyku TCL, který můžeme spustit příkazem `ns název_souboru.tcl`. Po spuštění se nám automaticky otevře okno nam. Jelikož jsme však nenadefinovali žádné uzly sítě, spoje a provoz v síti budete nam prázdný.

Po předchozích nezbytných krocích se konečně dostáváme k definování prvků sítě. Základním prvkem sítě je uzel, tento objekt vytvoříme příkazem $ns node, musíme však ještě dodat proměnou, která bude odkazovat na instanci tohoto objektu, celý příkaz tedy bude mít podobu `set n(0) [$ns node]`, analogicky můžeme vytvářet další uzly n(1) [$ns node], n(2) [$ns node] atd. Logickým krokem po definici uzlů je definování spojů mezi uzly. Spojíme např. uzel n(0) a n(1) pomocí duplexní linky s šířkou pásma 0,5 Mb, zpožděním 20ms a typem fronty drop tail. Toto provedeme následujícím příkazem `$ns duplex-link $n(0) $n(1) 0.5Mb 20ms DropTail`. Místo drop tail, což je vlastně fronta typu FIFO, NS2 umožňuje použít fronty např. CBQ (Class based queuing), RED (random early detection), SFQ(stochastic fair queing), WFQ (weighted fair queing), DRR (deficit round robin). Teď už jen zbývá nadefinovat přenos po duplexní lince mezi uzly n(0) a n(1). Nadefinujeme si např. provoz využívající spolehlivý transportní protokol TCP následujícím způsobem.

```
set tcp [new Agent/TCP]
$tcp set packetSize_ 100
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

Vytvořili jsme si tedy nejprve proměnnou tcp, která odkazuje na instanci objektu reprezentujícího zdroj dat, tento zdroj jsme přiřadili k uzlu n0, následně jsme vytvořili konzumenta dat TCPSink, kterého přiřadíme uzlu n1, přičemž jsme propojili zdroj dat tcp s konzumentem TCPSink dat. Dále jsme vytvořili generátor dat ftp, který jsme přiřadili zdroji dat tcp. Definice provozu, využívajícího nespolehlivý transportní protokol udp bude vypadat analogicky.

```
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set sink1 [new Agent/Null]
$ns attach-agent $n1 $sink1
$ns connect $udp $sink1
set cbr [new Application/Traffic/CBR]
```

```
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

Při definici udp provozu můžeme konzumenta dat Agent/Null, který přijaté pakety rovnou zahazuje, nahradit konzumentem dat Agent/LossMonitor nebo Agent/UDP. V obou případech provoz spustíme příkazem `$ns at 1.0 “ftp start”` a provoz zastavíme `$ns at 3.0 “ftp stop”`. Čísla v obou příkazech nám udávají čas zastavení nebo spuštění provozu (1.0 znamená 1 sekunda), za kterým následuje název provozu, který chceme spustit.

Z předchozího způsobu definice provozu můžeme vypozorovat jednu nevýhodu, budeme-li chtít definovat stejný provoz mezi více uzly, musíme pokaždé opisovat podobnou část skriptu jen s malými obměnami, proto je vhodné definovat provoz pomocí tzv. procedury, která může vypadat např. následovně.

```
proc ftp {odesilatel prijemce start} {
    global ns n
    set tcp($odesilatel) [new Agent/TCP]
    $tcp($odesilatel) set packetSize_ 1000
    $ns attach-agent $n($odesilatel) $tcp($odesilatel)
    set sink($prijemce) [new Agent/TCPSink]
    $ns attach-agent $n($prijemce) $sink($prijemce)
    $ns connect $tcp($odesilatel) $sink($prijemce)
    set ftp($odesilatel) [new Application/FTP]
    $ftp($odesilatel) attach-agent $tcp($odesilatel)
    $ftp($odesilatel) set type_ ftp
    $ns at $starttime "$ftp($odesilatel) start"
}
```

Tímto jsme vytvořili proceduru s názvem ftp s parametry odesilatel, příjemce a start (čas spuštění provozu). Proceduru zavoláme pomocí příkazu ftp 0 1 1.0, tímto jsme zahájili ftp provoz v čase 1 sekunda mezi uzly n(0) a n(1).

8.2 Konfigurace bezdrátového modelu sítě v NS2

Rozdíl mezi vytváření drátové a bezdrátové simulace je, že v případě bezdrátové simulace nevytváříme mezi uzly v síti spoje. Začátek skriptu tedy opět sestává z vytvoření objektu, který bude reprezentovat simulátor příkazem `set ns [new Simulator]`, vytvoříme soubory pro zapisování výsledků simulace `set tracef [open vysledky_simulace.tr w]`, `set namtr[open vysledky_simulace.nam w]` a nastavíme tyto soubory jako výstupní soubory simulace `$ns trace-all $tracef` , `$ns_ namtrace-all-wireless`.

V této chvíli se začínáme odlišovat od předchozí kapitoly, musíme definovat velikost plochy, na které se bude simulace odehrávat, vytvoříme tedy nový objekt s názvem plocha `set plocha [new Topography]` a nastavíme hranice této plochy `$plocha load_flatgrid 100 100`, v našem případě se jedná o rozlohu plochy o velikosti 100x100 metrů. Dále vytvoříme objekt god `set god_ [create-god (počet uzlů v síti)]`, který bude použit pro skladování informací o uzlech v síti např. počet uzlů v síti, informace o nejkratší cestě mezi dvěma uzly atd. Následuje vytvoření procedury konec, která bude stejná jako v předchozí kapitole. Dále nakonfigurujeme vlastnosti uzlů v síti před tím, než uzly začneme vytvářet.

```
$ns node-config -adhocRouting AODV \      směrovací protokol
     -llType LL \                         typ ll vrstvy
     -macType Mac/802_15_4 \              MAC typ
     -ifqType Queue/DropTail/PriQueue \   typ front
     -ifqLen 50 \                         velikost fronty
     -antType Antenna/OmniAntenna \       typ antény
     -propType Propagation/TwoRayGround \ model šíření signálu
     -phyType Phy/WirelessPhy/802_15_4 \  typ síťového rozhraní
     -topoInstance $plocha \              definice plochy
     -agentTrace OFF \
     -routerTrace OFF \
     -macTrace ON \
     -movementTrace OFF \
     -channel Channel/WirelessChannel \   typ přenosového kanálu
     -energyModel "EnergyModel" \         energetický model
```

```
-initialEnergy 1 \                počáteční energie v Joulech
-rxPower 0.045 \                  přijímací příkon ve W
-txPower 0.076 \                  vysílací příkon ve W
-sleepPower 0.00000006 \          spotřeba ve spánku ve W
-idlePower  0.0012 \              spotřeba v aktivním režimu W
```

Po nakonfigurování vlastností uzlů můžeme přikročit k vytváření uzlů a definovat jejich pozice na předem vytvořené ploše v našem případě o velikosti 100x100 metrů.

```
for {set i 0} {$i < 2) } {incr i} {
     set node_($i) [$ns_ node]
     $node_($i) random-motion 0
}
$node_(0) set X_ 50.0
$node_(0) set Y_ 20.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 70.0
$node_(1) set Y_ 85.0
$node_(1) set Z_ 0.0
```

Předchozím textem jsme vytvořili dva uzly, kterým jsme zakázali náhodný pohyb a nastavili jsme jejich souřadnice. Dále můžeme nastavit pohyb těchto uzlů.

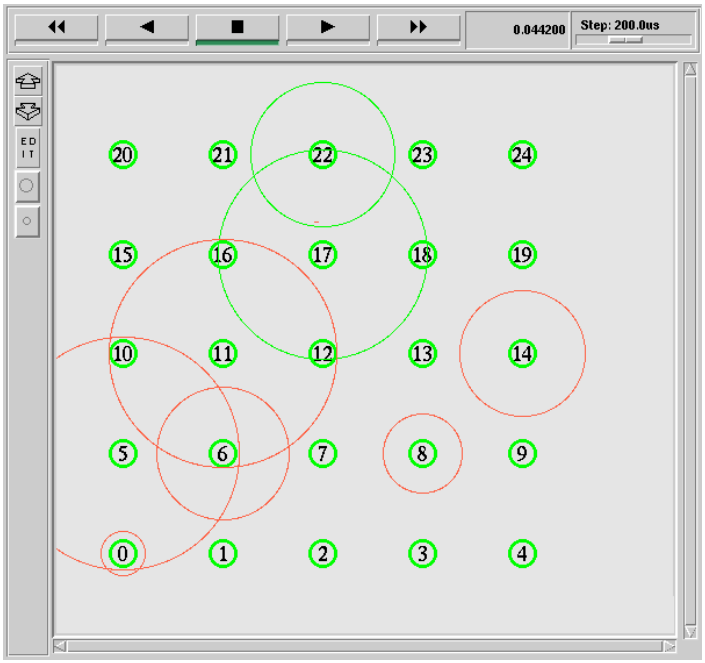
```
$ns at 5.0 "$node_(1) setdest 15.0 10.0 8.0"
```

Předchozí příkaz nám říká, že uzel číslo 1 se začne pohybovat v čase 5 sekundu od spuštění simulace směrem k souřadnicím x = 15, y = 10 rychlostí 8 m/s. Stejným způsobem můžeme nadefinovat pohyby i ostatních uzlů v síti.

Po definici velikosti plochy, vlastností uzlů, souřadnic uzlů a pohybu uzlů se dostáváme k definici provozu v síti, přičemž provoz se definuje stejným způsobem jako v předchozí kapitole 8.1.

9 Bezdrátová senzorová síť v NS2

V simulačním prostředí NS2 byl nakonfigurován model bezdrátové senzorové sítě (viz. obrázek 9.1), který sestává z 25 uzlů rozmístěných do mřížky ve vzdálenosti 20x20 metrů. Uzly jsou rozmístěny na ploše o velikosti 100x100 metrů.



Obr. 9.1: Bezdrátová senzorová síť

V tomto modelu byly zkoumány energetické nároky na jednotlivé uzly v síti (příjemce, směrovače, odesílatele) při různých způsobech přenosu. Data byla posílána sítí od uzlu číslo 22, vždy přes uzel číslo 17, který v simulaci figuroval jako směrovač, přičemž se měnil počet příjemců a velikost datových jednotek posílaných sítí. Pro konfiguraci energetického modelu v NS2 se vycházelo z hodnot spotřeby energie senzorů s mikroprocesorem mica2. Tyto senzory pracují v pásmu 2,4 GHz s maximální přenosovou rychlostí 250 kbit/s. Počáteční energii uzlů jsem nastavil na hodnotu 1 joule, z důvodů přehlednosti výsledků.

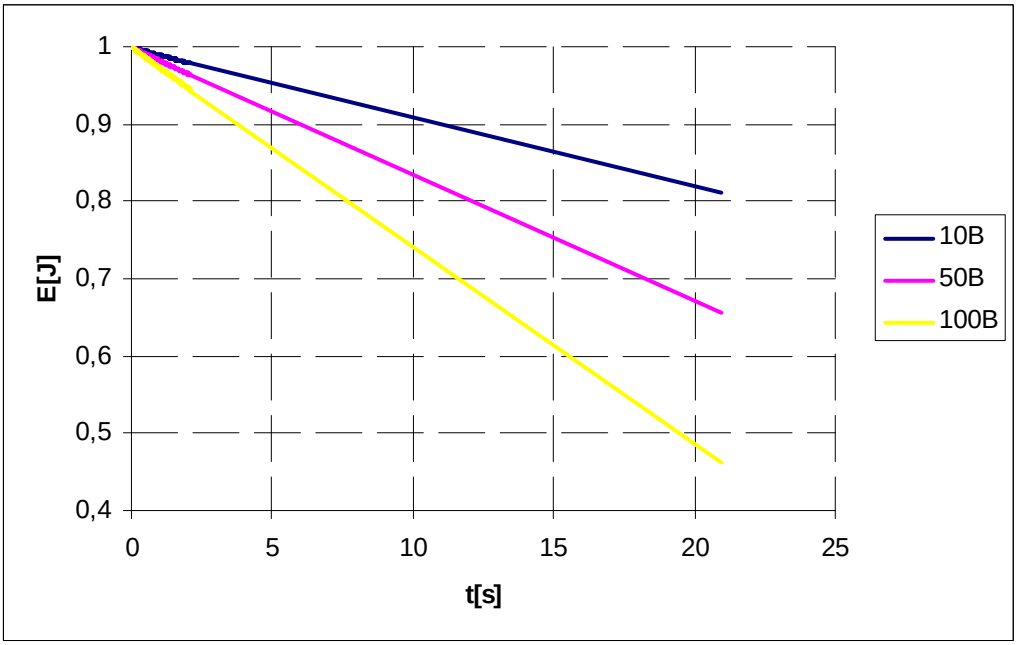
Tab. 9.1: Spotřeba energie senzorů s mikroprocesorem MICA 2 [9]

Operace	Spotřeba energie
Odeslání	25,4 mA
Příjem	15,1 mA
Aktivní režim	0,4 mA
Režim spánku	0,02 uA

V simulaci byla použita bezdrátová technologie 802.15.4 zigbee, tato technologie je totiž navržena s ohledem na spotřebu energie při komunikaci a je určena zejména pro bezdrátové senzorové sítě. Jako směrovací protokol byl použit unicast AODV, jedná se o ad-hoc směrovací protokol, který spojení sestavuje je-li potřeba a spojení je sestaveno jen na dobu nezbytně nutnou pro přenesení dat.

9.1 Spotřeba energie u odesílatele

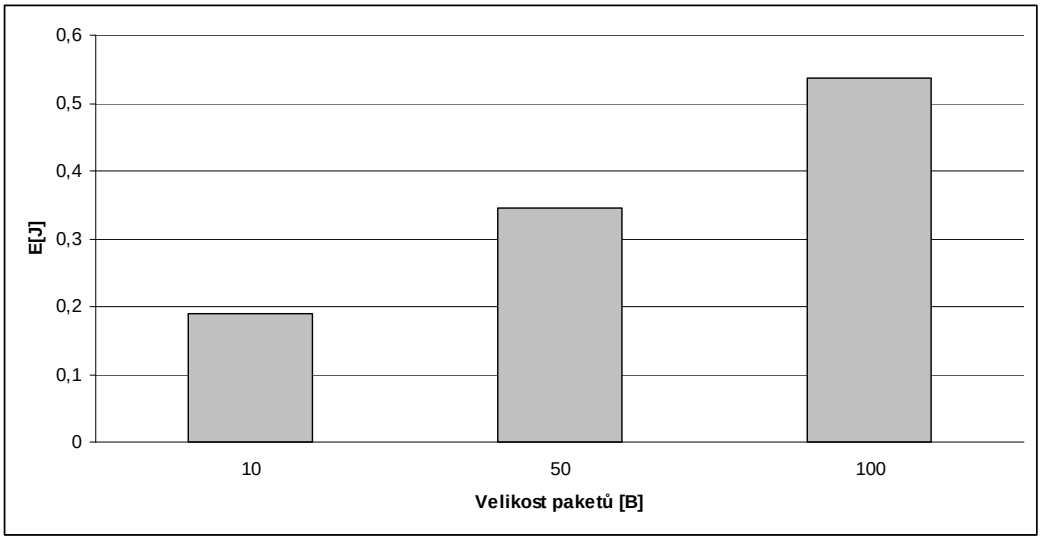
V první části výzkumu byla zkoumána energie, která je potřebná pro odeslání konstantního počtu paketů o různých délkách. V simulačním prostředí NS2 byla tedy postupně zvyšována velikosti paketů, přičemž při každé změně velikosti paketů, jsme provedli opětovné spuštění simulace, čímž byla postupně získávána potřebná data. V simulaci funkci odesílatele zastával uzel číslo 22, výsledky simulace jsou zobrazeny na obrázcích 9.2 a 9.3.



Obr. 9.2: Průběh spotřeby energie u odesílatele

Na obrázku 9.2 vidíme klesající hladiny energie uzlů při odesílání dat o velikostech 10 bajtů, 50 bajtů a 100 bajtů. Ve všech třech případech bylo odesláno 1000 paketů o výše zmíněných délkách pomocí protokolu UDP (User Datagram Protokol). Ve skutečnosti však musíme k odeslaným datům přičíst ještě velikost hlavičky UDP protokolu (20 bajtů) a velikost hlavičky MAC protokolu (7 bajtů), což byl v našem případě 802.15.4 zigbee.

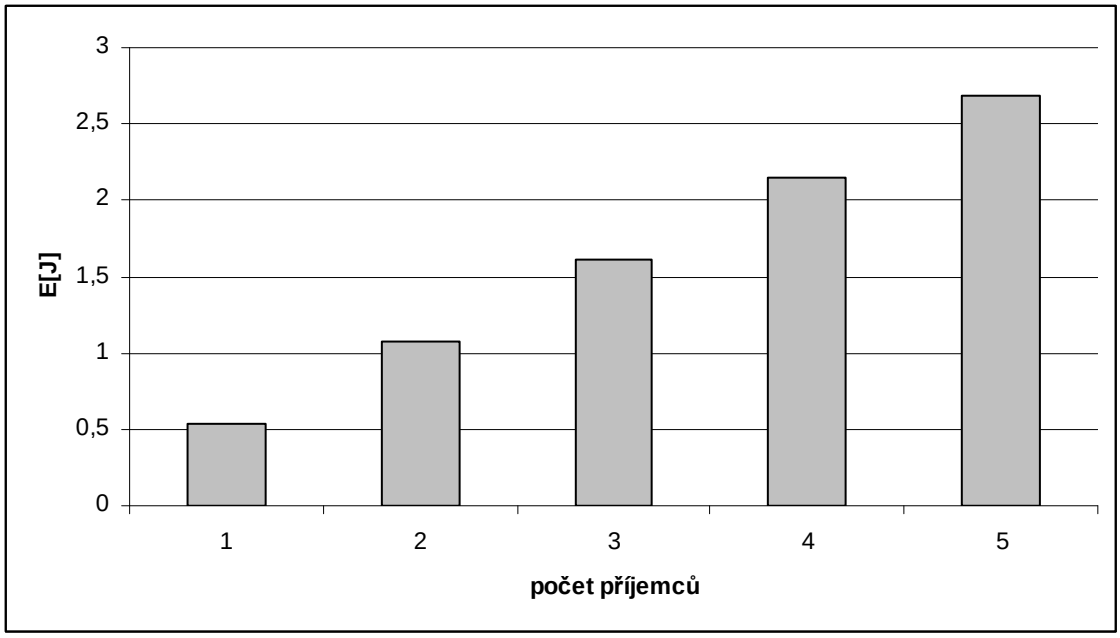
Odesílání větších paketů než 127 bajtů nemělo smysl uvažovat, protože se jedná o maximální velikost, kterou je schopen datagram definovaný dle standartu IEEE 802.15.4 přenést (7 bajtů záhlaví, 120 bajtů data). Při odesílání větších paketů by došlo pouze k jejich rozdělení na výše zmíněnou délku. Na obrázku 9.2 můžeme pozorovat, že energetické hladiny uzlů se začínají rozcházet až v čase 0,1 sekundy, to je způsobeno tím, že do této doby se hledá cesta v sítí pomocí směrovacího protokolu AODV, což probíhá všesměrovým vysíláním zpráv RREQ a následným zasíláním odpovědí RREP. Možná někomu přijde zvláštní, že při odesílání stejného počtu paketů o různých délkách je čas doručení všech paketů pokaždé stejný, to je způsobeno tím, že je v simulátor nastaven interval odesílání 50 paketů za vteřinu z důvodů lepší přehlednosti výsledků.



Obr. 9.3: Energie spotřebovaná pro odeslání paketů

Z obrázku 9.3 můžeme vyčíst, že nejvíce energie a to asi 0,5378 joulů tedy spotřeboval uzel, který odesílal pakety velikosti 100 bajtů, naopak nejméně energie asi 0,1895 joulů spotřeboval uzel, který odesílal pakety o nejmenší velikosti a to 10 bajtů. Méně energeticky náročné je tedy odeslání stejného počtu paketů o menší velikosti.

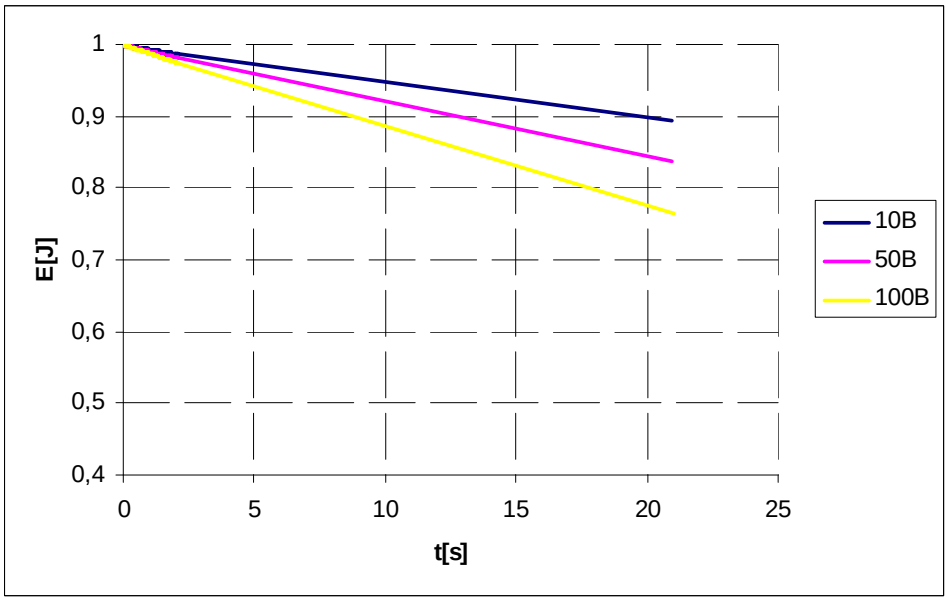
Na následujícím obrázku 9.4 je znázorněna vzrůstající spotřeba energie odesílatele při rostoucím počtu příjemců. V simulaci bylo odesláno 1 000 paketů o velikosti 100 bajtů postupně jednomu, dvěma, třem, čtyřem a pěti příjemcům. Spotřeba energie vzrůstala prakticky lineárně s rostoucím počtem příjemců, při jednom příjemci se spotřebovalo 0,53 joulů, při dvou 1,08 joulů, při třech 1,62 joulů atd. I kdyby všech 5 příjemců přijímalo stejná data, spotřeba u odesílatele by vzrůstala s počtem příjemců.



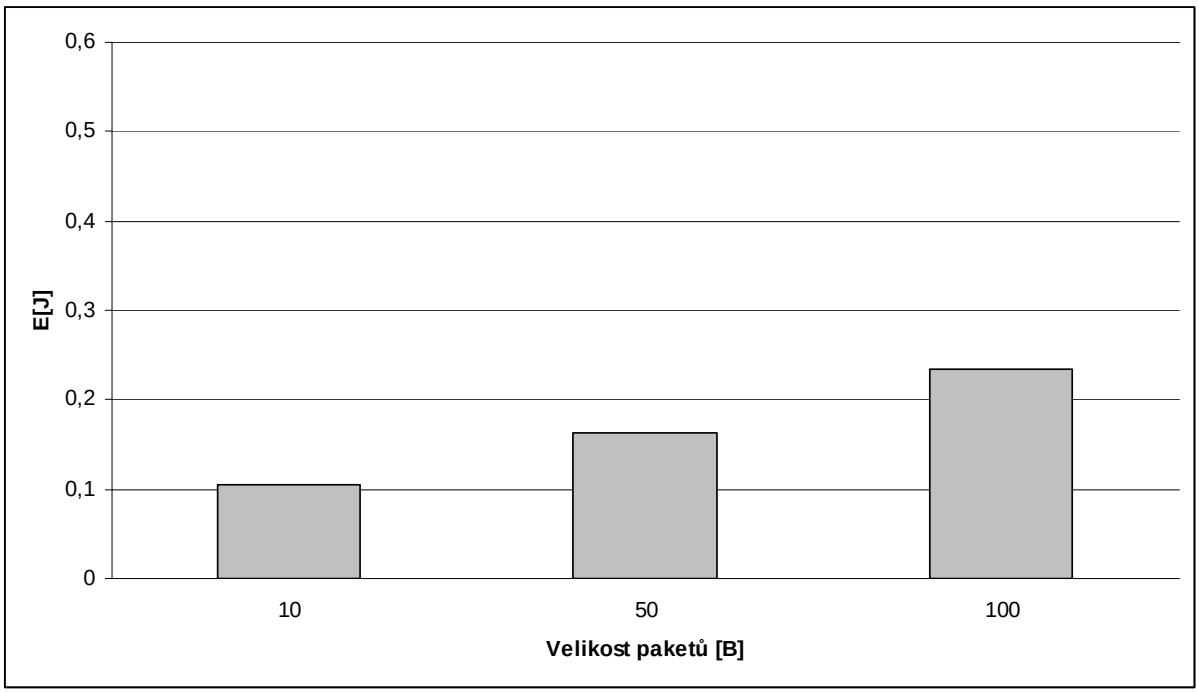
Obr 9.4: Spotřeba energie odesílatele při rostoucím počtu příjemců

9.2 Spotřeba energie u příjemce

V této kapitole byly zkoumány energetické nároky na příjemce v bezdrátové senzorové síti. Zajímala nás spotřeba energie při příjímání paketů o různých délkách, přičemž počet paketů posílaných sítí byl neměnný. Komunikace probíhala od uzlu číslo 22 směrem k uzlu 2, který zastával funkci příjemce.



Obr. 9.5: Průběh spotřeby energie u příjemce



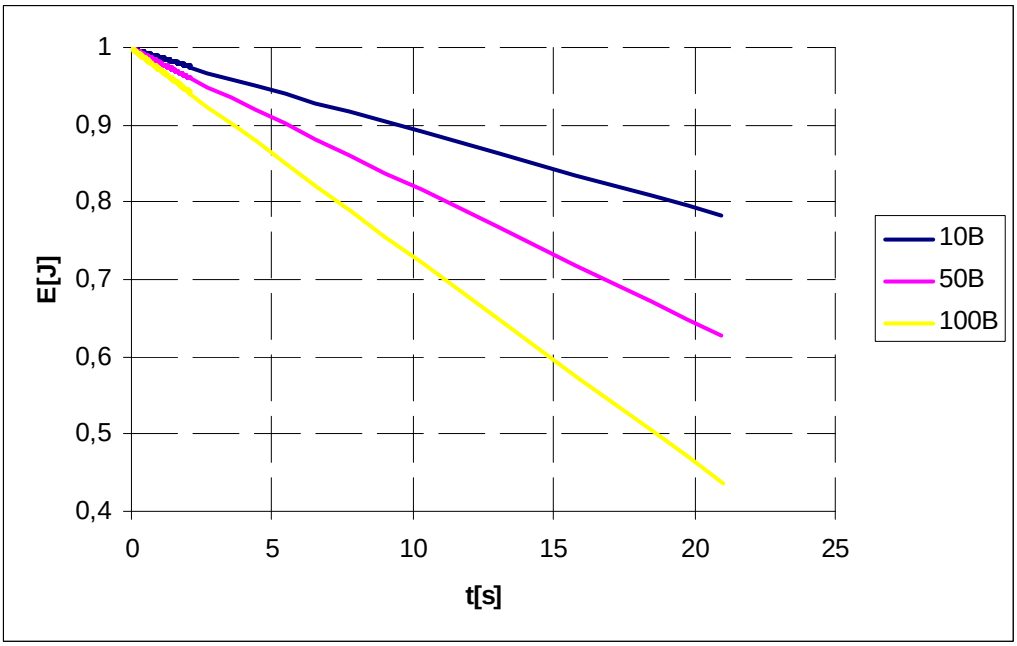
Obr. 9.6: Energie spotřebovaná pro příjem paketů

Na obrázku 9.5 vidíme spotřebu energie uzlů při příjmu 1000 paketů o délkách 10 bajtů, 50 bajtů a 100 bajtů. Jedná se o totožnou komunikaci jako v kapitole 9.2, tentokrát ale z pohledu příjemce. Při srovnání grafů na obrázcích 9.2 a 9.5 je patrné, že pro příjem paketů je spotřebováno menší množství energie než pro odeslání. Na začátku komunikace se opět hledá cesta sítí pomocí směrovacího protokolu AODV, proto opět asi do času 0,1 sekunda klesají energetické hladiny stejným tempem, od tohoto místa začíná nejvíce klesat energetická hladina uzlu při přijímání 100 bajtových paketů, z obrázku 8.4 vidíme, že tato hladina klesne po odeslání 1000 paketů o hodnotu 0,234 joulů, zatímco energetická hladina uzlu, který příjme 100 paketů o velikosti 10 bajtů klesne jen o 0,105 joulů. Při porovnání spotřeby energie při příjmu na obrázku 9.6 a energie pro odesílání na obrázku 9.3 vidíme, že spotřeba energie při přijímání paketů je asi poloviční, než spotřeba energie při odesílání paketů, přičemž při přijímaní paketů je opět energeticky méně náročné přijímat pakety o menší velikosti.

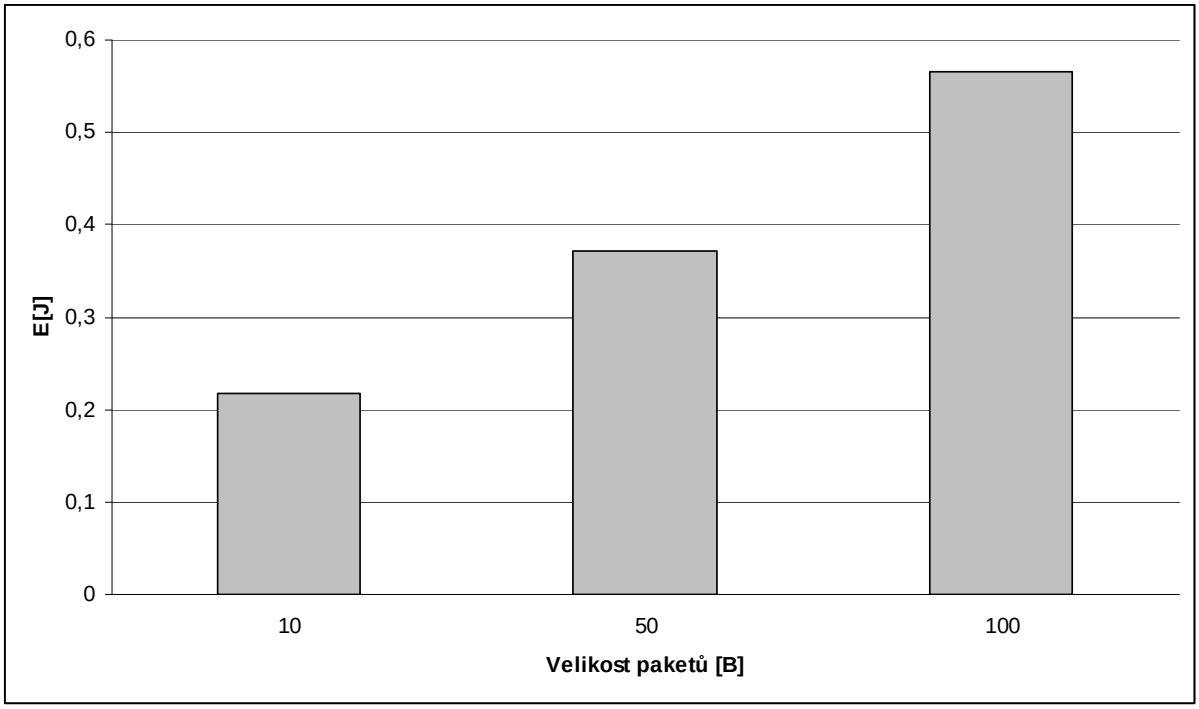
9.3 Spotřeba energie u směrovače

V předchozích dvou kapitolách jsme se zabývali spotřebou energie při odesílání a přijímání paketů v bezdrátové senzorové síti. V komunikačním řetězci nám chybí jeden článek, u kterého jsme zatím spotřebu energie nezkoumali a tím je směrovač. V této kapitole

tedy budeme zkoumat spotřebu energie při směrování paketů sítí. Síti budou směrovány pakety o různých velikostech, přičemž počet paketů bude opět konstantní.



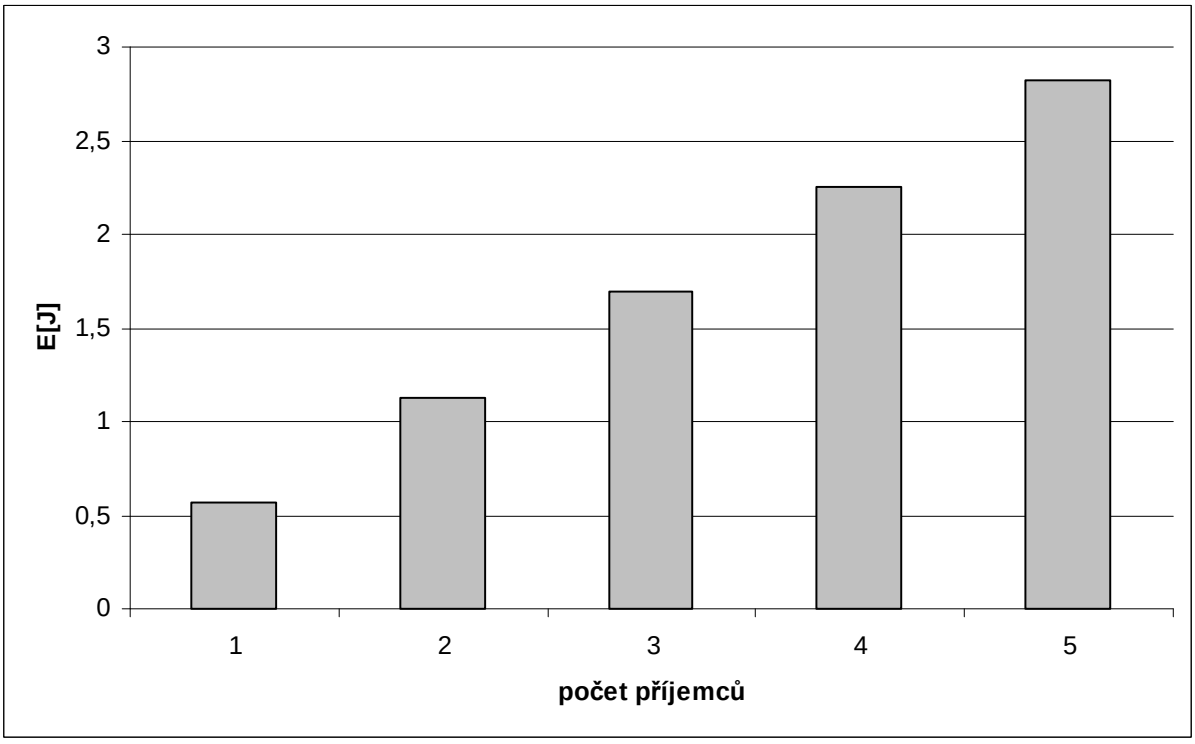
Obr 9.7: Průběh spotřeby energie u směrovače



Obr. 9.8: Energie spotřebovaná pro směrování paketů

Na obrázcích 9.7 a 9.8 je srovnání spotřeby energie uzlů při přeposílání 1000 paketů délek 10 bajtů, 50 bajtů a 100 bajtů. Opět vidíme, že uzel, který přeposílal 1000 paketů o velikosti 100 bajtů spotřeboval nejvíce energie a to asi 0,565 joulů, zatímco při přeposílání 100 paketů o velikosti 10 bajtů uzel spotřeboval pouze 0,2168. Při přeposílání paketů je tedy energeticky méně náročné přeposílat pakety o menší velikosti. Při srovnání grafů na obrázcích 9.2 až 9.8 vidíme, že nejméně energie se spotřebovává při přijímání paketů, zatímco nejvíce při jejich přeposílání.

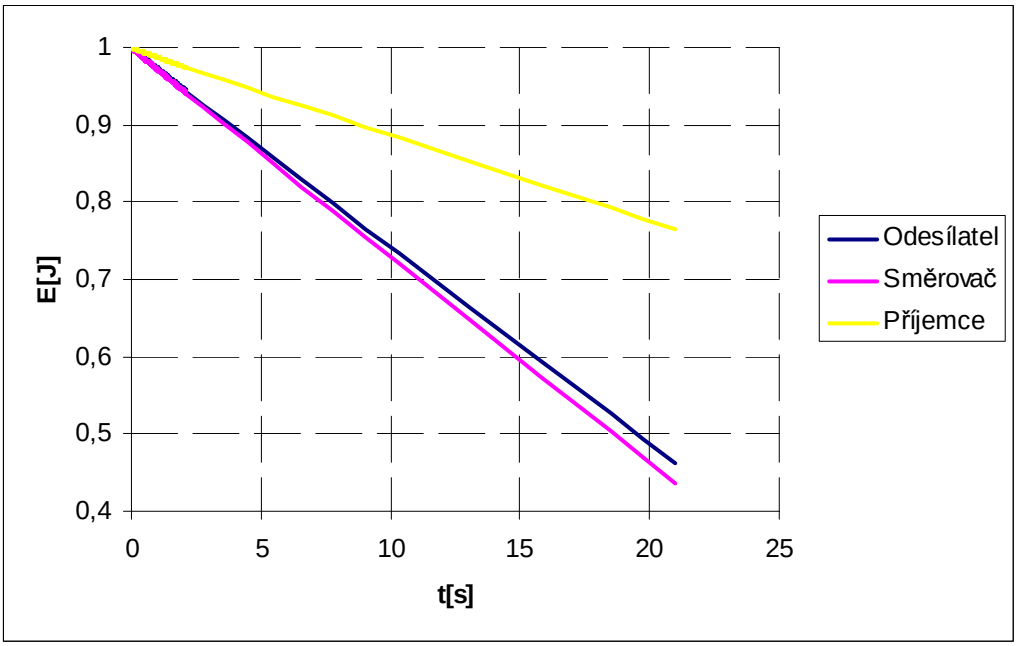
Na obrázku 9.9 můžeme pozorovat vzrůstající spotřebu energie uzlu, který zastává funkci směrovače, při přeposílání 1 000 paketů o velikosti 100 bajtů pomocí UDP protokolu. Vidíme, že spotřeba energie stoupá lineárně s rostoucím počtem příjemců, zatímco při jednom příjemci činila spotřeba 0,565 joulů, při dvou je to asi dvojnásobek 1,129 joulů a při třech trojnásobek 1,1694 joulů atd. Spotřeba by vzrůstala i v případě, pokud by se přeposílala stejná data pěti různým příjemcům.



Obr 9.9: Spotřeba energie směrovače při rostoucím počtu příjemců

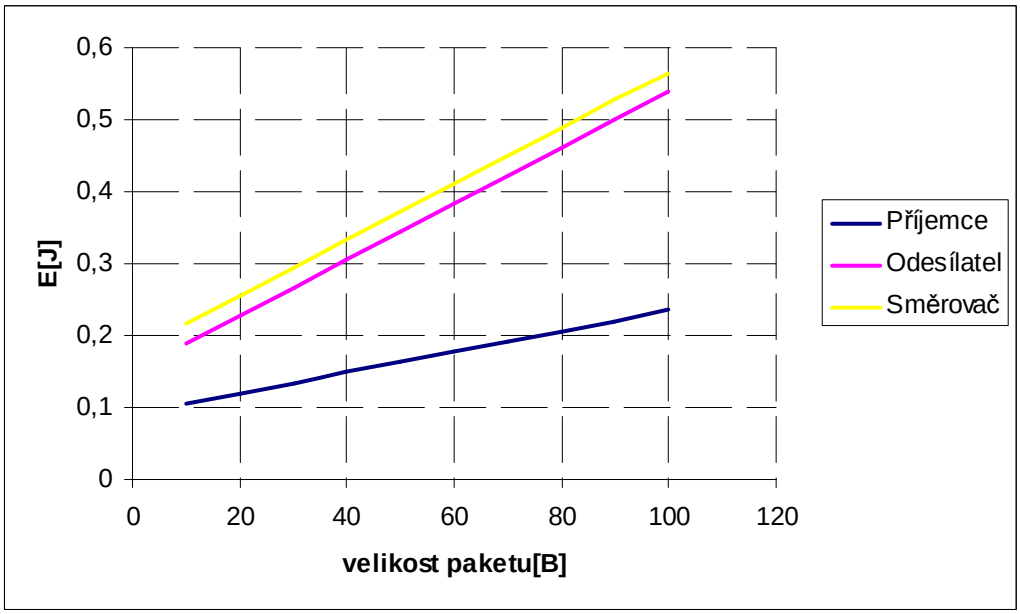
9.4 Srovnání spotřeby energie u jednotlivých uzlů

V této části výzkumu bylo provedeno srovnání spotřeby energie u jednotlivých uzlů v síti podílejících se na komunikace (odesílatel, směrovač, příjemce). Srovnání bylo provedeno v závislosti na posílání konstantního množství dat sítí, ale také v závislosti na posílání dat různých délek v rozmezí 10 bajtů až 100 bajtů. Výsledky výzkumu jsou uvedeny na obrázcích 9.10 až 9.12.



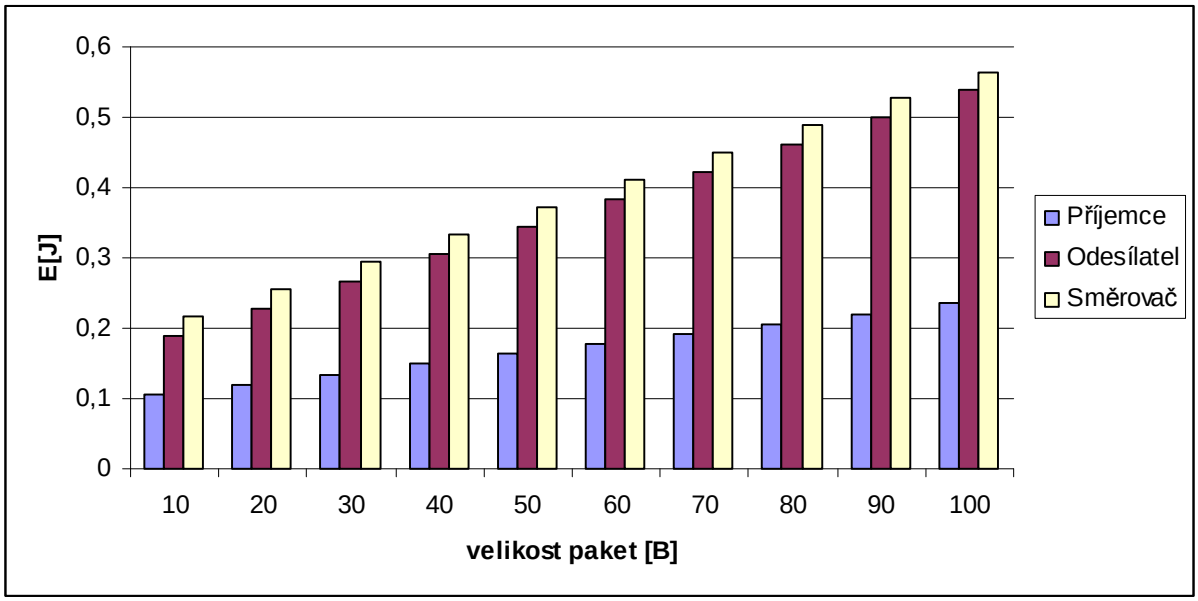
Obr. 9.10: Průběh spotřeby energie uzlů podílejících se na komunikaci

Na obrázku 9.10 vidíme srovnání spotřeby energie uzlů, kteří se podílejí na komunikaci v bezdrátové senzorové síti. Síti bylo posíláno 1000 paketů o velikosti 100 bajtů, opět UDP protokolem. Směrovač tedy spotřeboval pro směrování 1000 paketů o velikosti 100 bajtů asi 0,565 joulů. Pro odeslání stejného množství dat odesílatel spotřeboval 0,532 joulů, přičemž při příjmu těchto se spotřebovalo 0,234 joulů. Vidíme, že největší spotřebu energie při komunikaci má dle předpokladů směrovač, který při přeposílání paketů zastává funkci příjemce i odesílatele, ovšem rozdíl mezi spotřebou energie odesílatele a směrovače 0,033 joulů není nijak markantní. Nejmenší spotřebu energie má dle očekávání příjemce.



Obr. 9.11: Závislost spotřeby energie na velikosti odesílaných paketů

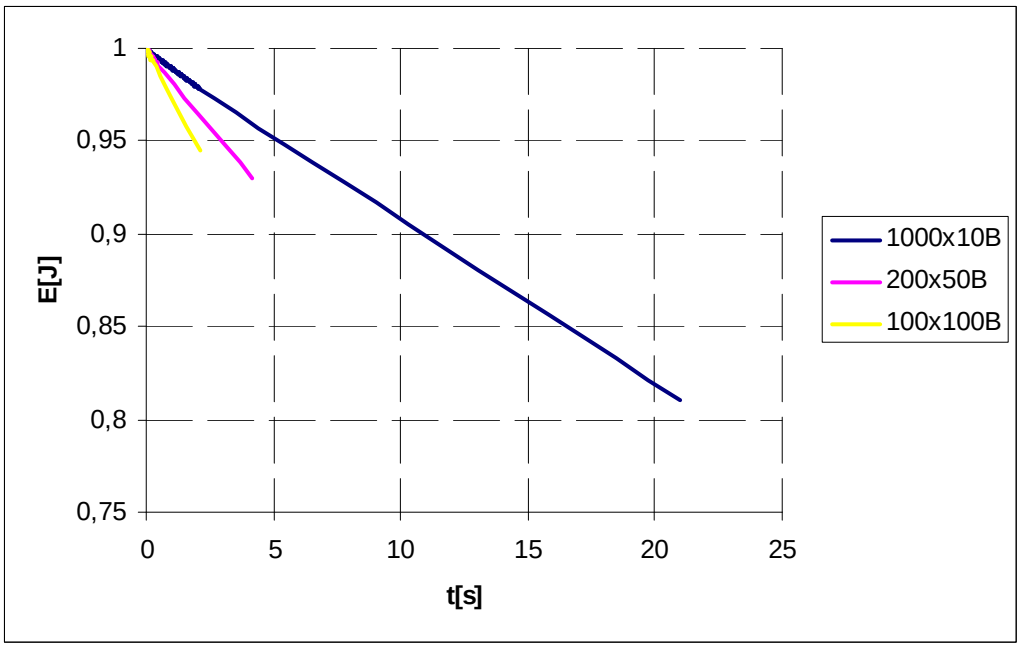
Na následujícím obrázku 9.11 je znázorněna závislost spotřeby energie jednotlivých uzlů podílejících se na komunikaci na velikosti posílaných paketů. Sítí bylo posíláno 1000 paketů o velikostech 10 až 100 bajtů pomocí protokolu UDP. Z obrázku 9.12 je patrné, že spotřeba energie vzrůstala s velikostí paketů, přičemž nejmenší spotřebu energie má příjemce a největší směrovač. Rozdíl mezi spotřebou odesílatele a směrovače je opět poměrně malý.



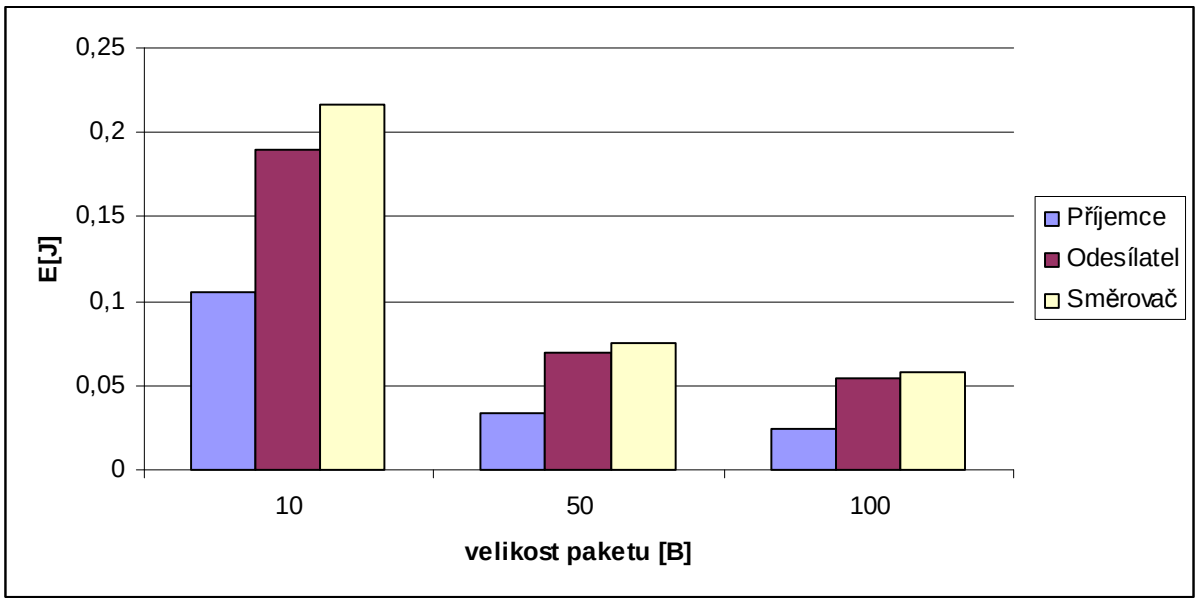
Obr. 9.12: Srovnání spotřeby energie v závislosti na velikosti paketů

9.5 Spotřeba energie v závislostí na posílání konstantního množství dat

Při tomto experimentu bylo bezdrátovou senzorovou síti odesláno konstantní množství dat, přičemž tyto data byla pokaždé odeslána různě velkými pakety. Cílem bylo zjistit, který z těchto způsobů odesílání dat je energeticky méně náročný. V simulátoru byl nastavován maximální počet paketů, který se bude odesílat v závislosti na velikosti paketů. Výsledky simulace jsou uvedeny na obrázcích 9.13 a 9.14.



Obr. 9.13: Průběh spotřeby energie odesílatele při odesílání dat o velikosti 10 000 bajtů



Obr. 9.14: Spotřeba energie při odesílání dat o konstantní velikosti 10 000 bajtů

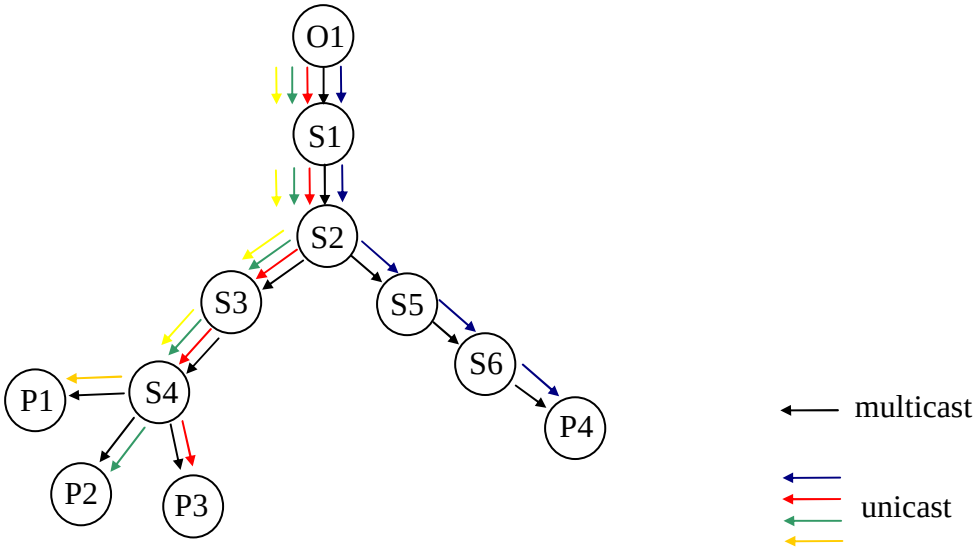
Z výsledků v předchozích kapitolách 9.2 až 9.6 by se mohlo zdát, že je energeticky méně náročné posílat data síti menšími pakety, jelikož poslání 1000 paketů délky 10 bajtů je daleko méně energeticky náročné než poslání stejného množství paketů o velikosti 100 bajtů. Musíme si však uvědomit, že při odesílání stejného množství paketů o menší velikosti přeneseme sítí od odesílatele směrem k příjemci také menší množství dat.

Na obrázku 9.13 je znázorněna situace, kdy přenášíme sítí data o velikosti 10 000 bajtů pomocí UDP protokolu. Pro 1 000 bajtů dat musíme odeslat 1000 paketů o velikosti 10 bajtů, 200 paketů o velikosti 50 bajtů a 100 paketů o velikosti 100 bajtů. V prvním případě spotřebujeme při přenosu 0,19 joulů, v druhém 0,071 joulů a v třetím 0,054 joulů. Nejméně energeticky náročné je tedy posílat data pakety o co možná největší velikosti, jelikož každý paket má hlavičku a ta je vždy stejně velká, v našem případě činila hlavička 27 bajtů. Pokud odešleme 1000 paketů o velikosti 10 bajtů, odešleme v hlavičkách paketů 27 000 bajtů, přičemž při odesílání 100 paketů o velikosti 100 bajtů je to 10krát méně, proto je spotřeba energie při odesílání dat pakety o menší velikosti několikanásobně větší.

Na obrázku 9.13 je zobrazen pouze průběh spotřeby energie při odesílání dat o konstantní délce 10 000 bajtů, spotřeba energie dalších uzlů podílejících se na komunikaci v bezdrátové senzorové (směrovač, příjemce, odesílatel) je zobrazena na obrázku 9.14. Z tohoto obrázku je patrné, že při směrování 1 000 paketů o velikosti 10 bajtů se spotřebovala energie o velikosti 0,22 joulů, pro příjem těchto paketů se spotřebovalo 0,11 joulů. Přičemž pro směrování stejného množství dat, která byla posílána pakety o velikosti 100 bajtů, jejichž počet byl 10krát menší se spotřebovalo 0,07 joulů a pro příjem 0,025 joulů, což je opět asi 10 krát méně. Při posílání 10 000 bajtů je energeticky méně náročné posílat data většími pakety pro všechny uzly, které se podílejí se na komunikaci.

9.6 Spotřeba energie při hromadné komunikaci

Pokud potřebujeme stejná data poslat určité skupině příjemců, je výhodnější tyto data odeslat multicast komunikací než unicast komunikací. Multicast komunikací jsou data sítí poslána pouze jednou, přičemž o následné doručení se starají směrovače, síť je tedy daleko méně vytížená, jelikož se pošle od odesílatele několikanásobně menší množství dat. Srovnání unicast a multicast komunikace vidíme na obrázku 9.15.



Obr. 9.15: Srovnání unicast a multicast komunikace ve WSN síti

Na obrázku 9.15 je zobrazena komunikace mezi odesílatelem O1 a příjemci P1 až P4. Předpokládáme, že potřebujeme odeslat stejná data všem příjemcům P1 až P4. Pokud by tato komunikace probíhala jako unicast, musíme data odeslat nejprve příjemci P1, pak příjemci P2, P3 a P4, tudíž od odesílatele O1 přes směrovače S1 a S2 odesíláme stejná data čtyřikrát, třikrát přes směrovače S3 a S4 a jednou přes S5 a S6, jelikož se zde nachází pouze jeden příjemce.

Pokud bychom použili pro odeslání dat multicast, byla by data odeslána od odesílatele směrem k příjemcům pouze jednou, přičemž směrovače S1 až S6 přeposílají tyto data také pouze jednou. Při posílání dat multicastem tedy odešle odesílatel O1 čtyřikrát méně dat, směrovače S1, S2 přeposílají čtyřikrát méně dat a S3, S4 přeposílají třikrát méně dat než při unicast komunikaci. Dle vzorců 7.1 a 7.2 v kapitole 7 vidíme, že spotřeba energie uzlu ve WSN síti je závislá na počtu odeslaných a přijatých bitů, tudíž pokud při multicast komunikaci dochází k odeslání menšího množství dat, dochází též k menší spotřebě energie uzlů. Spotřebu energie při komunikaci typu multicast nebylo možné ověřit pomocí simulačního prostředí NS2 z důvodu nekompatibility balíčku multicast pro bezdrátové sítě s novějšími verzemi programu NS2.

10 Závěr

Bezdrátové senzorové sítě neboli wireless sensor networks mají velký potenciál do budoucna, i když uplatnění by jistě našli v některých oborech již dnes, právě proto jsou v dnešní době tyto sítě terčem mnoha výzkumů. WSN mohou být použity v aplikacích např. pro měření tlaku, teploty, vlhkosti, průtoku, pro dohled nad určitou oblastí, jako kouřová nebo pohybová čidla. Přičemž počet prvků WSN může být desítky, stovky i tisíce v závislosti na rozloze plochy, kterou máme v úmyslu monitorovat. Jedním z největších problémů, se kterým se u WSN potýkáme je spotřeba energie, jelikož je většina prvků sítě WSN napájena pomocí bateriových článků, přičemž požadujeme výdrž těchto bateriových článků až v řádů několika let. Proto se snažíme co nejvíce snížit spotřebu energie u jednotlivých prvků WSN sítě.

Spotřebu energie můžeme v první řadě ovlivnit výběrem prvků WSN, kde můžeme dle spotřeb energie senzorových obvodů, mikroprocesoru, přijímací a vysílací části komunikačních obvodů uváděných výrobcem vybrat co možná nejšetrnější prvky pro naši WSN síť. Spotřeba energie při komunikaci bývá uváděna spotřebou pro vyslání jednoho bitu tx v závislosti na vysílacím výkonu a spotřebou energie pro příjem jednoho bitu rx. Úsporu energie při komunikaci můžeme tedy ovlivnit omezením maximálního dosahu vysílače a omezením počtu přenášených bitů. Při omezení vysílacího výkonu nebude uzel schopen komunikovat na větší vzdálenosti, proto budeme muset komunikaci koncipovat skokově, což znamená, že při komunikaci mezi dvěma vzdálenými uzly, budou do komunikace také zapojeny mezilehlé uzly, které budou směrovat pakety od odesílatele směrem k příjemci.

V simulačním prostředí network simulátor 2 byl nakonfigurován model bezdrátové senzorové sítě s přenosovou technologií 802.15.4 a unicast směrovacím protokolem AODV. Pro konfiguraci energetického modelu v NS2 se vycházelo z hodnot spotřeby energie senzorů s mikroprocesorem mica2 (tabulka 9.1). V kapitole 9 jsou uvedeny výsledky simulace, při zkoumání spotřeby energie u jednotlivých uzlů při komunikaci v závislosti na velikosti přenášených paketů. Sítí bylo od odesílatele směrem k příjemci posíláno 1 000 paketů o velikostech 10 až 100 bajtů pomocí UDP protokolu. Na obrázcích 9.2 až 9.7 jsou zobrazeny spotřeby energie uzlů podílejících se na komunikaci. Nejmenší spotřebu energie měl příjemce, pak následoval odesílatel a největší spotřebou disponoval směrovač, přičemž spotřeba energie vzrůstala lineárně s velikostí přenášených paketů. Dle obrázků 9.4 a 9.9 vidíme, že spotřeba energie u odesílatele i směrovače vzrůstala s počtem příjemcům, a to i v případě, pokud od odesílatele přes směrovač k příjemcům byla posílána stejná data.

Dále byla zkoumána spotřeba energie při posílání konstantního množství dat o velikosti 10 000 bajtů pomocí UDP protokolu pakety o velikosti 10bajtů, 50bajtů a 100bajtů. Z obrázků 9.13 a 9.14 je patrné, že energeticky méně náročné bylo přenést 10 000 bajtů pomocí menšího počtu větších paketů, jelikož hlavičky paketů mají stejnou velikost ať jsou pakety jakkoliv velké, tudíž při přenášení konstantního množství dat pakety o menší velikosti se přenáší více nadbytečných bajtů v hlavičkách. Pro odeslání 10 000 bajtů pomocí paketů o velikosti 10 bajtů musíme odeslat 1 000 paketů, přičemž při posílání 10 000 bajtů pomocí paketů o velikosti 100 bajtů se odešle pouze 100 paketů.

Pokud máme v síti více příjemců, kteří čekají na stejná data, je výhodnější k přenosu použít komunikaci typu multicast. Na obrázku 9.15 vidíme srovnání komunikace typu unicast a multicast. Pokud data odesíláme více příjemcům pomocí unicast komunikace, spotřeba energie odesílatele a směrovačů vzrůstá s počtem příjemcům, jak ukazují obrázky 9.4 a 9.9. Při multicast komunikaci dochází k poslání dat od odesílatele k příjemcům pouze jednou, od odesílatele přes směrovače je tedy posláno menší množství bitů než při unicast komunikaci. Dle vzorců 7.1 a 7.2 je spotřeba energie uzlu WSN sítě závislá od počtu odeslaných a přijímaných bitů, tudíž pokud při použití multicast komunikace je od odesílatele přes směrovače posláno menší množství bitů, spotřebuje se také menší množství energie pro jejich odeslání a směrování. Spotřebu energie při komunikaci typu multicast nebylo možné ověřit pomocí simulačního prostředí NS2 z důvodu nekompatibility balíčku multicast pro bezdrátové sítě s novějšími verzemi programu NS2, přičemž starší verze neměla podporu bezdrátových sítí.

11 Seznam použité literatury

[1] DEERING S.: Host extensions for IP multicasting ,RFC1112, August 1989

[2] WAITZMAN D., PARTRIGE C., DEERING S.: Distance Vector Multicast Routing Protocol, RFC1075 November 1988.

[3] PERKINS C., BELDING-ROYER E., DAS S.: Ad hoc On-Demand Distance Vector (AODV) Routing, RFC3561, July 2003.

[4] FENNER W.: Internet Group Management Protocol, Version 2, RFC 2236, November 1997.

[5] CAIN B., DEERING S., KOUVELAS I., FENNER B., THYAGARAJAN A.: Internet Group Management Protocol, Version 3, RFC 3376, October 2002.

[6] KUORILEHTO M., KOHVAKKA M., SUHONEN J., HAMALAINEN P., HANNIKAINEN M., HAMALAINEN T.: Ultra-low energy wireless sensor networks in practice, John Wiliey & Sons, LTD, 2007.

[7] SILVA R., SILVA J., SIMEK M., BOAVIDA F.: Why should multicast be used in WSNs, June 2008.

[8] FALL K., VARDHAN K.: The ns Manual, Berkeley, 2003.

[9] Chiang M.: Energy optimization in wireless sensor networks, VDM Verlag Dr. Müller, 2007.

11 Abecední přehled použitých zkratek, veličin a symbolů

Přehled použitých zkratek

Zkratka	Anglický popis	Český popis
AODV	Ad hoc On-Demand Distance Vector	Ad hoc směrovací protokol
FFD	Full function device	Zařízení s plnou funkcí
IGMP	Internet Group Management Protocol	Protokol pro správu multicast skupin
IP	Internet Protocol	Internet Protokol
MAC	Media Access Control	Přístup k médiu
MAODV	Multicast Ad hoc On-Demand Distance Vector	Multicast Ad hoc směrovací protokol
NS2	Network Simulator 2	Simulační program
PIM	Protocol Indepentend Multicast	Multicast směrovací protokol
RERR	Route Error	Zpráva o ztrátě spojení
RFD	Reduced Function Device	Zařízení s omezenou funkčností
RREP	Route Replay	Odpověď na RREQ
RREQ	Route Request	Hledání spojení
UDP	User Datagram Protocol	Protokol pro nespolehlivý přenos
WSN	Wireless Sensor Network	Bezdrátová senzorová síť