VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

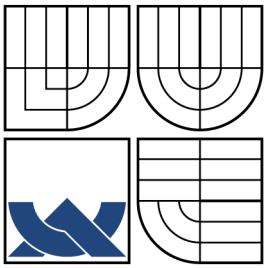
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

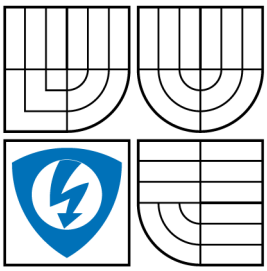
ANALÝZA MODELŮ SMĚROVACÍCH PROTOKOLŮ OLSR A AODV PRO MANET SÍTĚ V PROSTŘEDÍ OPNET MODELER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. TOMÁŠ MACHATA
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ANALÝZA MODELŮ SMĚROVACÍCH PROTOKOLŮ OLSR A AODV PRO MANET SÍTĚ V PROSTŘEDÍ OPNET MODELER

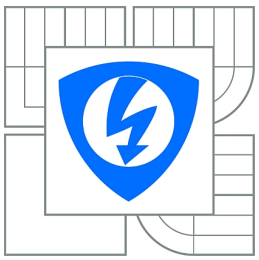
ANALYSIS OF MANET ROUTING PROTOCOLS OLSR AND AODV IN OPNET MODELER SIMULATION ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ MACHATA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ HOŠEK
SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Tomáš Machata	***ID:***	98260
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Analýza modelů směrovacích protokolů OLSR a AODV pro MANET sítě v prostředí OPNET Modeler

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku MANET sítí se zaměřením zejména na procesy směrování. Proveďte teoretický rozbor směrovacích protokolů OLSR a AODV. V praktické části diplomové práce se detailně zaměřte na protokol AODV. V simulačním prostředí OPNET Modeler vytvořte model sítě MANET a nakonfigurujte do něj směrovací protokol AODV. Dále se pokuste provést optimalizaci parametrů směrovacího protokolu za účelem dosažení lepších charakteristik síťového provozu. V prostředí OPNET Modeler prostudujte procesní model protokolu AODV a doplňte jej o definici nového typu zprávy, díky níž bude možné přenášet uživatelsky definované informace. Všechny dosažené výsledky uveďte přehledně v závěrečné práci.

DOPORUČENÁ LITERATURA:

[1] ILYAS, M.: The Handbook of Ad Hoc Wireless Networks. Boca Raton: CRC Press, 2003, ISBN: 0-8493-1332-5.
[2] MOHAPATRA, P., KRISHNAMURTH, S.: Ad Hoc Networks: Technology and Protocols. Boston: Springer Press, 2005, ISBN: 0-387-22689-3.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Jiří Hošek

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá problematikou MANET sítí a je zaměřena zejména na procesy směrování. Největší pozornost je věnována směrovacím protokolům AODV a OLSR. Tyto protokoly jsou zde podrobně popsány.

Cílem této práce je vytvoření simulačního modelu MANET sítě v prostředí OPNET Modeler. V tomto modelu je nakonfigurován AODV protokol a za účelem dosažení lepších charakteristik síťového provozu je provedena optimalizace parametrů směrovacího protokolu.

Dále je v tomto prostředí prostudován a rozšířen procesní model AODV protokolu o nový typ zprávy, který umožňuje přenášet aktuální přenosovou rychlost síťového rozhraní MANET stanice. Aktuální přenosová rychlost jednotlivých stanic je vyčtena ze statistik. Každá stanice v pravidelných intervalech posílá novou zprávu sousedním uzlům. Uzel po přijetí nové zprávy uloží informace do souboru.

KLÍČOVÁ SLOVA

Ad-hoc sítě, AODV, MANET, OLSR, OPNET Modeler, směrovací protokol

ABSTRACT

The thesis deals with MANET networks and it focuses on the routing process. Most attention is paid to the routing protocols AODV and OLSR. These protocols are described in the detail.

The aim of the thesis is to create a simulation model of MANET network in OPNET Modeler environment. In this model the AODV protocol is configured. In order to achieve improved characteristics of the network traffic the routing protocol parameters are optimized.

Furthermore the process model of AODV protocol in this environment is studied and extended by a new type of message, which allows a transfer of current transmission speed of MANET station network interface. Current transmission rate of stations is retrieved from the statistics. Every station periodically sends a message to neighboring nodes. The node stores the information into the file when a new message arrives.

KEYWORDS

Ad-hoc Networks, AODV, MANET, OLSR, OPNET Modeler, routing protocol

MACHATA, Tomáš *Analýza modelů směrovacích protokolů OLSR a AODV pro MANET sítě v prostředí OPNET Modeler*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011. 87 s. Vedoucí práce byl Ing. Jiří Hošek

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Analýza modelů směrovacích protokolů OLSR a AODV pro MANET sítě v prostředí OPNET Modeler“ 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.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Jiřímu Hoškovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.
Dále bych chtěl poděkovat firmě OPNET Technologies, Inc. za poskytnutí licence k programu OPNET Modeler.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Využívání Internetu v posledních letech rapidně vzrostlo, a to zejména zásluhou multimediálního obsahu. Zatím je stále převládající způsob připojení k internetu pomocí kabelu nebo optických vláken, ale zvyšuje se počet uživatelů, kteří chtějí mít stálý přístup bez ohledu na to, kde se právě nacházejí. Charakter zařízení, které bude mít všudypřítomný přístup k informacím, umožňují bezdrátové sítě představující nejjednodušší řešení pro jejich vzájemné propojení.

Pokroky v bezdrátové komunikaci umožňují nové zásadní komunikační modely: samo-organizující informační a komunikační systémy. V tomto novém síťovém modelu jsou uživatelé mobilních zařízení součástí sítě a musí spolupracovat na zajištění funkčnosti celé sítě. Takové systémy jsou označovány jako mobilní ad-hoc sítě (MANET) nebo také jako méně-infrastrukturované bezdrátové sítě.

Mobilní ad-hoc sítě (Mobile Ad-hoc Networks, dále jen MANET) jsou takové sítě, kde jsou jednotlivé uzly mobilní a jejich vzájemné propojení se může neustále měnit. V takovém druhu sítě je velice důležité vyřešit problémy, které mohou nastat při směrování a zajištění bezpečnosti.

Tato diplomová práce je zaměřena právě na MANET sítě. Cílem této práce je teoreticky popsat procesy směrování v MANET sítích, a to zejména z pohledu AODV a OLSR směrovacích protokolů. V praktické části této práce je vytvořen model MANET sítě v simulačním prostředí OPNET Modeler. V tomto modelu je nastaven AODV protokol a je zde zkoumán vliv jednotlivých parametrů směrovacího protokolu na vlastnosti síťového provozu (zpoždění, zatížení a další). V další části je prostudován a rozšířen procesní model AODV protokolu o nový typ zprávy. Tato zpráva umožňuje přenášet hodnotu aktuální přenosové rychlosti bezdrátového síťového rozhraní MANET stanice. Jednotlivé stanice v pravidelných intervalech zasílají novou zprávu sousedním uzlům. Stanice, které obdržely nový typ zprávy, zprávu zpracují a uloží vyčtené informace do souboru.

V první kapitole je uvedena stručná charakteristika MANET sítí. Druhá kapitola je zaměřena na směrování v MANET sítích. Tato kapitola se zabývá pouze problematikou unicastového směrování. Třetí kapitola je o OLSR protokolu a ve čtvrté kapitole je podrobně popsán AODV protokol.

Pátá kapitola obsahuje popis simulačního prostředí OPNET Modeler. Tato kapitola také obsahuje podrobný popis implementace AODV protokolu v tomto prostředí. V šesté kapitole je uveden postup tvorby modelu MANET sítě. V sedmé kapitole je zmínka o tvorbě modelu, který umožňuje vytvoření a příjem zpráv pomocí datové struktury ICI. Tato kapitola slouží zejména k porozumění práce s procesními a uzlovými modely. Poslední kapitola obsahuje podrobný návod jak rozšířit AODV protokol o nový typ zprávy.

1 MANET SÍTĚ

Myšlenka bezdrátového, mobilního Internetu není nic nového. V roce 1969 byly přeneseny první pakety v síti ARPANET. Ministerstvo obrany okamžitě pochopilo potenciál přepojování paketů bezdrátovou technologií k propojení mobilních uzlů na bitevním poli. V 70. letech se zrodil projekt DARPA Packet Radio, který stanovil pojem ad-hoc bezdrátové sítě. Jedná se o technologie, které umožňují nevázaný bezdrátový provoz v prostředích bez pevné nebo mobilní infrastruktury (např. bojiště, místo přírodní katastrofy, ...) nebo tam, kde není dostačující infrastruktura nebo by náklady na její výstavbu byly enormní.

Termín „ad-hoc“ označuje dočasné síťové spojení mezi dvěma rovnocennými uzly. Dříve bylo toto spojení zřízeno pouze pro zvláštní nebo improvizované služby přizpůsobené aplikacím. V dnešní době se takovéto spojení používá i pro veřejné účely. Protokoly jsou laděny na určité aplikace (např. zřízení videokonference mezi záchrannými jednotkami, vysílání video streamu, ...). Aplikace mohou být mobilní a prostředí se může dynamicky měnit. V důsledku tohoto musejí být ad-hoc protokoly schopny samo-nastavitelnému přizpůsobení se prostředí, změn provozu a poslání. Z těchto vlastností vyplývá, že tato technologie je extrémně flexibilní, pružná a přesto robustní.

Z důvodů mobilní bez-infrastrukturové povahy ad-hoc sítě jsou kladeny nové požadavky na návrh. První je samo-nastavitelnost (adres, směrování, ...). Na aplikační úrovni uživatelé ad-hoc sítě často komunikují a spolupracující jako tým (např. policie, hasiči, záchranné mise, ...). Tyto aplikace tedy vyžadují efektivní skupinové komunikace (multicasting) pro data a provoz v reálném čase. Navíc mobilita podněcuje velké množství lokálně založených služeb, které neexistují v pevně spojeném Internetu. Složitostí návrhu mobilních ad-hoc sítí se zabývaly generace výzkumných pracovníků od 70. let. Ale až díky velkým pokrokům v radiokomunikaci byly zaznamenány velké úspěchy jak ve vojenské, tak i v civilní sféře. Na první pohled se může zdát, že se ad-hoc sítě nehodí do představy „infrastrukturované sítě a Internetu“, na které většina komerčních aplikací spoléhá. To je částečně důvod, proč ad-hoc technologie měla těžký přechod na obchodní scénáře a každodenní používání obyčejnými lidmi [6].

1.1 Charakteristika MANET sítí

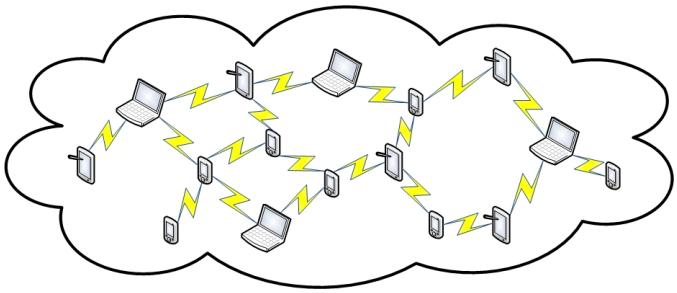
MANET sítě (obr. 1.1) jsou systém bezdrátových mobilních platforem (např.: bezdrátové routery, bezdrátové zařízení, ...), označované zjednodušeně jako „uzly“, které se volně a libovolně pohybují. Uzly mohou být umístěny na letadlech, lodích, nákladních automobilech, autech, dokonce i na lidech nebo velmi malých zařízení.

MANET je autonomní systém mobilních uzlů. Systém může pracovat samostatně nebo může být propojen s pevnou sítí pomocí bran.

MANET uzly jsou vybaveny bezdrátovými vysílači a přijímači využívajícími antény, které mohou být všesměrové (broadcast), směrové (poin-to-point), případně řiditelné nebo kombinované. V daném okamžiku, v závislosti na pozicích uzlů a jejich vysílačů, dosahu příjemců, úrovni přenosového výkonu a sdílení společných kanálových interferencí, je bezdrátové spojení mezi uzly v podobě multi-hop nebo ad-hoc sítě. Ad-hoc topologie se může měnit s časem, jak se uzly pohybují a přizpůsobují přenosovým a přijímacím parametrům [11].

MANET má několik hlavních charakteristických vlastností:

- Dynamická topologie - uzly se mohou volně a libovolně pohybovat, a tak se topologie sítě, která je většinou více skoková, může měnit náhodně a rychle v nepředvídatelných časech. Jednotlivé trasy se mohou skládat z obousměrných a jednosměrných spojení.
- Omezená šířka pásma a proměnná kapacita spojů - bezdrátové spojení má i nadále výrazně nižší kapacitu než pevně propojené spojení. Navíc propustnost bezdrátové komunikace po zohlednění vlivu vícenásobného přístupu, slábnutí signálu, šumu a interferencí je často mnohem menší než maximální radiová přenosová rychlost. Relativně nízká kapacita spojů může způsobit jejich zahlcení či přetížení. To může být problém, pokud mobilní síť rozšiřuje síť s pevnou infrastrukturou, kde mají uživatelé větší požadavky na šířku pásma a kapacitu spojů.
- Energicky omezený provoz - některé nebo všechny uzly v MANET síti využívají ke své funkci baterie nebo jiné vyčerpatelné zdroje energie. Proto je u těchto uzlů nejdůležitějším kritériem pro návrh systému úspora energie.
- Omezená fyzická bezpečnost - mobilní bezdrátové sítě jsou obecně náchylnější k fyzickým bezpečnostním hrozbám než pevně propojené sítě. Existuje větší možnost odposlechu, falšování zpráv a útokům pro odmítnutí služby (denial of service). Stávající technologie zabezpečení jsou často uplatňovány v rámci bezdrátových sítí, aby se snížilo bezpečnostní riziko [11].



Obr. 1.1: MANET síť

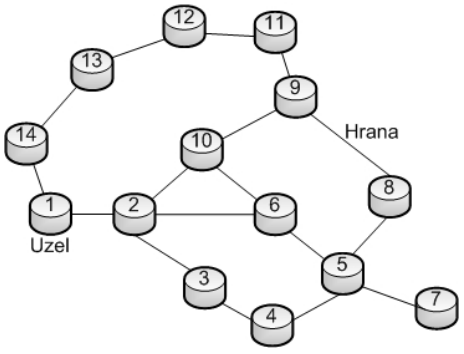
2 SMĚROVÁNÍ

Pojem MANET odkazuje na sadu bezdrátových mobilních uzlů, které mohou komunikovat a pohybovat se ve stejnou dobu. Žádná pevně daná infrastruktura není schopna umožnit takovou komunikaci. Všechny uzly spolupracují na úkolu směrování paketů do místa určení. Toto je důležité, protože každý uzel sítě je schopen komunikovat pouze s těmi uzly, které jsou umístěny uvnitř jeho přenosového dosahu (poloměru) R. Zatímco zdrojový uzel S a cílový uzel D MANET sítě mohou být umístěny ve vzdálenosti mnohem větší než je přenosový dosah R. Pokud chce uzel S poslat paket uzlu D, musí tento paket překonat několik mezilehlých uzlů. Z tohoto důvodu MANET sítě patří do skupiny multi-hop (více-skokových) bezdrátových sítí.

Tato kapitola je zaměřena na problém unicastového směrování, které je zodpovědné za směrování paketů z jednoho zdrojového uzlu do jednoho cílového uzlu, a uvádí hlavní techniky používané v návrhu směrovacích protokolů pro takové víceskokové bezdrátové prostředí. Pokud se cíl směrování skládá ze sady uzlů, je nutné použít multicastové směrování. Touto problematikou se v této práci nebudeme dále zabývat a zaměříme se pouze na unicastového směrování. Nejprve si stručně ukážeme, jak lze problém směrování vyjádřit v abstraktním prostředí. Síť je většinou modelována jako graf [2]:

$$G = \langle V, E \rangle \,, \tag{2.1}$$

kde $V = \{1, \ldots, n\}$ je množina uzlů (směrovačů) grafu a $E \subseteq V \times V$ je množina hran (spojů) (viz. obr. 2.1).



Obr. 2.1: Graf představující síť

Při použití technologie MANET, prvky patřící vrcholu množiny V odpovídají mobilním stanicím (předpokládá se fixní počet) a hrany patřící množině hran E odpovídající bezdrátovým spojům v síti. Bezdrátové spojení mezi dvěma uzly i a j je navázáno, když je fyzická vzdálenost PD(i,j) menší nebo rovna přenosovému dosahu

R (typická hodnota pro R je 250 m). Kvůli náhodné mobilitě uzlů, jsou nová bezdrátová spojení neustále vytvářena a stávající spojení jsou rušena a odstraňována. Toto vede k takzvané náhodné jednotce grafu.

Změna v množině E grafu se nazývá topologická změna řízená mobilitou uzlu. Topologická změna se také vyskytuje v pevně spojených sítí, ale v tom případě jsou to spíše následky poruch spojení než normální pracovní režim sítě.

Cesta $P(S, D)$ z uzlu S do D je specifikována jako posloupnost uzlů [2]:

$$P(S, D) \equiv \langle N_0, N_1, \ldots, N_K \rangle \,, \tag{2.2}$$

$$\text{kde } N_0 = S, N_K = D, N_i \neq N_j, (N_i, N_{i+1}) \in E(\text{pro } i \neq j, 0 \leq i \leq k-1).$$

Délka cesty $|P(S, D)|$ je počet spojů (skoků), jimiž prochází pakety cestou do cílového uzlu, tedy $|\langle N_0, N1, \ldots, N_K \rangle| = K$. Úlohu směrování je možně rozdělit do dvou dílčích úkolů:

1. Výpočet cesty $P(S, D)$,
2. Posílání datových paketů po zjištěné trase.

Cesty jsou počítány pomocí distribuovaných algoritmů běžících na síťové vrstvě, která zahrnuje určitou formu údržby cesty po změně topologie. Údržba aktuální topologie sítě může být prováděna pomocí periodické aktualizace trasy a případně odesíláním aktualizací při změně síťové topologie. Takto by měl algoritmus sledovat změny topologie a důsledně obnovovat cesty. Také může nastat situace (velmi pravděpodobné), že existuje více cest mezi uzly S a D a směrovací protokol musí vypočítat nejlepší cestu P^* s ohledem na některé metriky. Například: obecně používaná metrika je délka cesty. V tomto případě je nejkratší cesta P^* počítána pomocí vzorce [2]:

$$P^*(S, D) = min_P \{|P(S, D)|\} \,. \tag{2.3}$$

Uzel $N_i \neq D$, který příjme paket adresovaný pro uzel D (cílový), přepošle tento paket na spojení propojené s dalším uzlem N_{i+1} cesty $P(S, D)$. Jsou dvě hlavní techniky pro takovéto přeposlání paketů:

1. Každý uzel má přístup do lokální směrovací tabulky se záznamy pro všechny uzly v síti s uvedeným dalším skokem na uzel. Další skok není závislý na zdrojovém uzlu S.
2. Každý uzel ukládá celou cestu do záhlaví paketu, takže informaci o next-hop uzlu (N_{i+1}) se uzel dozví od samotného paketu. Vzhledem k tomu, že cesta je ukládána v paketu od zdroje, nazývá se tato metoda směrování od zdroje [2].

2.1 Unicast směrovací protokoly pro MANET sítě

Unicast směrovací protokoly pro MANET sítě lze rozdělit do dvou kategorií podle toho, jestli zdrojová stanice před zahájením přenosu dat zná cestu k cílové stanici.

1. **Spojově orientované směrovací protokoly** - při použití těchto směrovacích protokolů musí být cesta od zdroje k cíli známa již před začátkem samotného přenosu dat. Tato kategorie protokolů je nejrozšířenější a patří do ní většina protokolů používaných v klasických sítích. Spojově orientované směrovací protokoly lze dále rozdělit podle způsobu přístupu k sestavení spojení do tří skupin: proaktivní, reaktivní a hybridní.
2. **Nespojové protokoly** - při využití těchto protokolů nemusí být známo spojení mezi zdrojovou a cílovou stanicí. Směrování datových paketů je založeno na geografické poloze stanic. Do této skupiny patří například hierarchické směrovací protokoly [2, 6].

2.1.1 Proaktivní protokoly

Proaktivní směrovací protokoly jsou velice podobné tradičným protokolům používaným v pevně spojených sítích, mezi které patří Distance Vector (DV) protokoly, Link State (LS) protokoly a smíšené LS a DV protokoly. Tyto tradiční protokoly jsou pouze přizpůsobeny pro mobilní prostředí při zachování jejich charakteru.

Pojem proaktivní odkazuje na schopnost protokolu propočítat všechny možné cesty bez ohledu na jejich efektivní využití. Proaktivní protokoly reagují na změny topologie, i když jimi není ovlivněn přenos paketů. Při použití tohoto směrovacího protokolu je směrovací tabulka uzlu vytvořena ihned po připojení do sítě a poté periodicky aktualizována.

Velkou výhodou těchto protokolů je, že cesta k cíli je okamžitě k dispozici a tak nedochází ke zpoždění, když aplikace potřebuje posílat pakety. V některých případech jsou tyto vlastnosti velice důležité, zejména pro interaktivní aplikace. Hlavní mechanismy využívané v proaktivních protokolech jsou:

- větší množství informací o topologii uložených v každém uzlu (zabránění smyček a zrychlení konvergence protokolu),
- dynamická velikost aktualizací trasy nebo četnost aktualizací,
- optimalizace záplavy (zaslání zprávy všem uzlům),
- kombinace DV a LS vlastností.

Mezi proaktivní protokoly patří Destination Sequenced Distance Vector (DSDV), Optimized Link State Routing (OLSR), Fisheye State Protocol (FSR) a Wireless Routing Protocol (WRP) [2].

2.1.2 Reaktivní protokoly

Reaktivní směrovací protokoly používají odlišný přístup pro směrování v mobilním prostředí. Tento přístup je označován jako směrování na požádání a vyznačuje se tím, že stanice zjišťuje cestu k cílovému uzlu pouze tehdy, pokud chce s tímto uzlem zahájit datovou komunikaci. Dále odstraňuje tradiční směrovací tabulky v uzlech, a proto sleduje změny topologie sítě, aby mohl aktualizovat směrovací informace.

Směrovací protokoly na požádání vypočítají cestu před přenosem dat. Pokud není datový provoz generován uzly, pak směrovací činnost zcela chybí. Z tohoto důvodu jsou označovány jako reaktivní protokoly.

Reaktivní protokol je charakterizován následujícími procedurami, které se používají ke správě cest:

- objevení cesty,
- údržba cesty,
- zrušení cesty (nepovinný).

Předávání dat se provádí podle dvou hlavních technik:

1. zdroj směrování,
2. skok za skokem (Hop-by-hop).

Procedura objevení uzlu je založena na mechanismu dotaz-odpověď, který je šířen záplavově. Cíl je nakonec dosažen pomocí dotazu a alespoň jedné odpovědi. Hledání cesty je spouštěno asynchronně na požádání, když je potřeba vysílat datové pakety a žádná cesta k cíli není známa. Hledání trasy není nutné spouštět pro každý datový paket, protože je pravděpodobné, že zjištěná cesta bude platná po celou dobu vysílání na stejné místo určení.

Jako výsledek vyhledávání cest získají síťové uzly nový „směrovací stav“, který obsahuje cesty zjištěné během hledání. Jednotlivé informace o směrování jsou spravovány pomocí údržbových mechanismů, dokud jsou používány nebo explicitně smazány.

Někteří zástupci směrovacích protokolů se vyznačují různými údržbovými mechanismy a používají různé techniky záznamu trasy směrování (např.: dočasné směrovací tabulky, paměti tras, logickou strukturu a další). Do skupiny reaktivních protokolů patří Dynamic Source Routing protocol (DSR), Ad Hoc On Demand Distance Vector routing (AODV), Adaptive Distance Vector (ADV), Temporally Ordered Routing Algorithm (TORA) a Associativity Based Routing protocol (ABR) [2].

2.1.3 Hybridní protokoly

Hybridní směrovací protokoly jsou kombinací proaktivních a reaktivních směrovacích protokolů. Snaží se kombinovat výhody obou protokolů. Tedy především snížení zpoždění potřebné k zjištění nové trasy a režie protokolu. Ústředním prvkem tohoto

typu protokolu je rozdělení sítě do určitých zón. Mezi hybridní protokoly patří Zone Routing Protocol (ZRP) a Hazy Sighted Link State (HSLS) [2].

3 OPTIMIZED LINK STATE ROUTING

Optimized Link State Routing (OLSR) patří do skupiny proaktivních decentralizovaných směrovacích protokolů, které se vyznačují tím, že informace potřebné ke směrování k jednotlivým uzlům jsou zjišťovány ještě před vznikem požadavku. Což vede k minimalizaci zpoždění při nově vzniklé komunikaci, ale na druhou stranu vzniká větší zatížení sítě režijními informacemi, a také vyšší výpočetní zatížení jednotlivých uzlů. OLSR byl vyvinut týmem Hipercom v INRIA a jeho specifikace je dostupná v RFC 3626 [13].

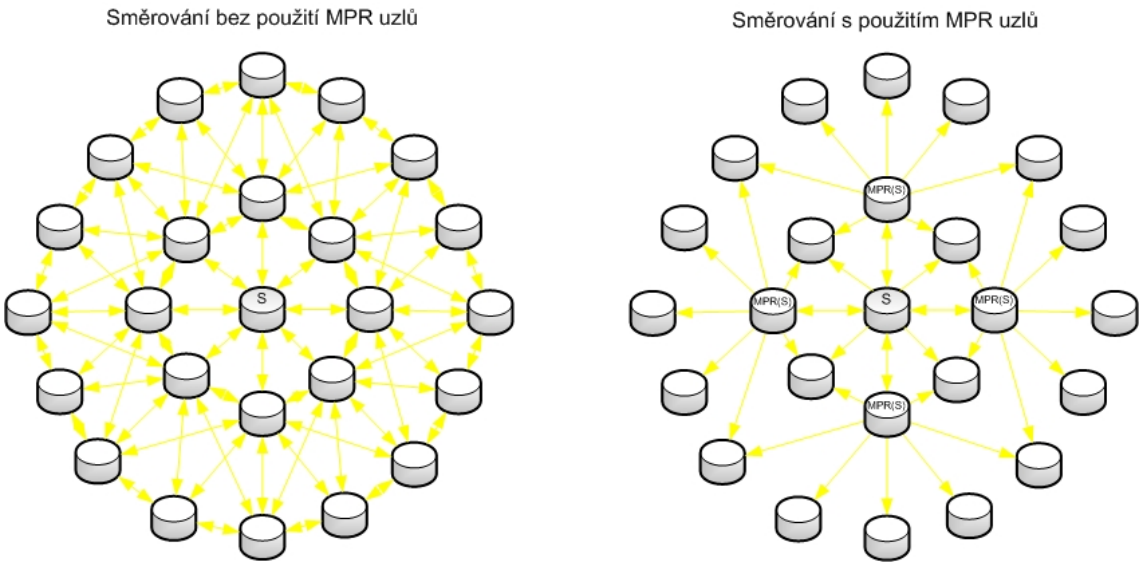
Jak je již z názvu patrné, OLSR představuje optimalizaci klasického protokolu stavu linky (LS protokol) upraveného pro potřeby mobilních ad-hoc sítí. Využívá klasický algoritmus při hledání nejkratší cesty založený na metrice počtu skoků. Hlavní myšlenkou OLSR je použití tzv. násobných uzlů pro přenos (MPR - Multipoint relay) k zaplavení sítě efektivní cestou z důvodů snížení duplicitních paketů ve stejné oblasti (viz. obr. 3.1). Protokol také vybírá obousměrná spojení pro účely směrování, aby se vyhnul problémům s přenosem paketů přes jednosměrná spojení. Každý uzel i vybírá ze sousedních jedno-skokových (one-hop) uzlů minimální (nebo téměř minimální) sadu multipoint relay uzlů, které označuje jako MPR(i). Uzly MPR(i) mají následující vlastnosti: každý uzel v symetrickém dvou-skokovém (two-hops) sousedství uzlu i musí mít symetrické spojení směrem k MPR(i). MPR sady dovolují efektivní realizaci zaplavení. Když uzel i chce odeslat zprávu, pošle zprávu pouze na uzly MPR(i), které postupně pošlou zprávy na jejich MPR uzly a tak dále.

Multipoint Relay selektor sady (MPR selector set) uzlu j je tvořen řadou sousedů, které jsou označeny jako MPR. Každý uzel pravidelně aktualizuje své MPR výběrové sady pomocí zaplavovacích technik a speciálního druhu kontrolních zpráv označovaných jako Topology Control (TC) zprávy. Použitím TC zpráv oznamuje uzel do sítě, že má dostupnost k uzlům s MPR selektorem sady (je to jejich uzel posledního skoku). TC zpráva je označena sekvenčním (pořadovým) číslem, které se zvětšuje pokaždé, když se MPR selektor sady mění.

Chceme-li zvýšit reakce na změny topologie, které jsou omezeny režií protokolu, je potřeba snížit na minimum časový interval mezi dvěma následujícími TC zprávami přenosu. Také, pokud má uzel prázdný MPR selektor sady, nemusí vytvářet žádné TC zprávy. Avšak pokud se MPR selektor sady vyprázdní, bude stále posílat TC zprávy po dobu, než se zruší předchozí TC zprávy. Informace získané z TC zpráv jsou použity k vytvoření topologie sítě a poté i směrovacích tabulek [2].

Uzly běžící pod OLSR přijímají následující tři datové struktury založené na:

- snímání informací v sousedství uzlů,
- snímání informací o topologii,
- směrovacích tabulkách.

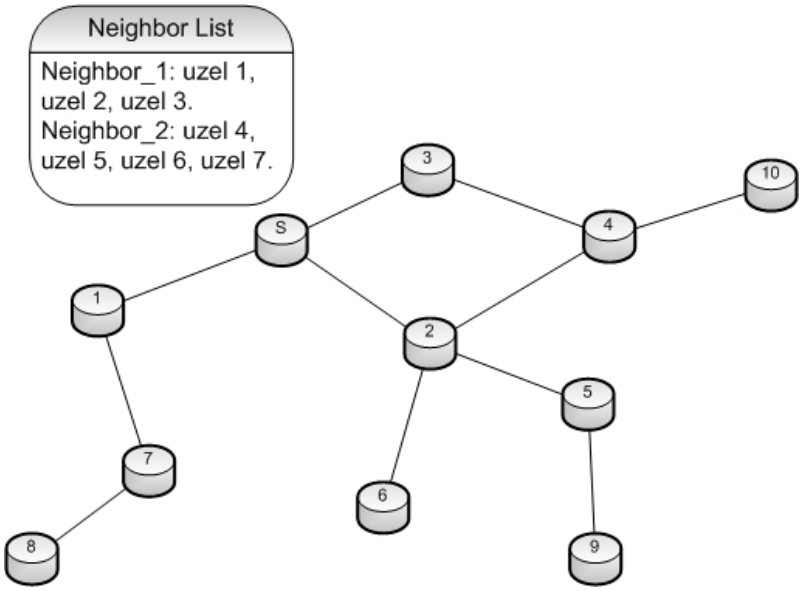


Obr. 3.1: Rozdíl směrování s a bez použití MPR

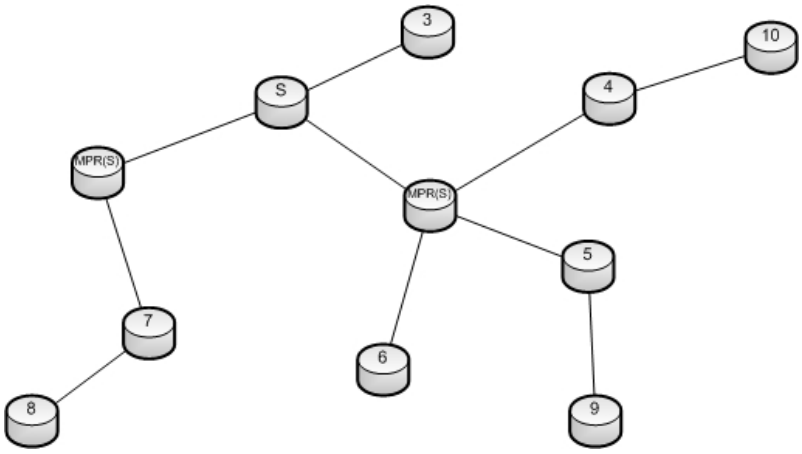
3.1 Princip protokolu OLSR

Princip směrovacího protokolu OLSR lze rozdělit do následujících čtyř kroků:

1. Zjištění sousedů (Neighbor sensing): Každý uzel pravidelně posílá zprávy typu HELLO s přednastavenou hodnotou TTL (Time To Live, Doba přežití) na hodnotu 2. V každé HELLO zprávě je uložena tabulka, kam se zaznamenávají hodnoty vlastních sousedů a jejich sousedů. Tedy každý uzel udržuje seznam (Neighbor List) všech uzlů do vzdálenosti dvou skoků (obr. 3.2).
2. Vybrání MPR uzlů (MPR Selection): MPR uzly se vybírají ze sousedních uzlů a musí umožnit spojení s uzlem, který je dva skoky vzdálený. Pokud více uzlů umožňuje spojení se všemi uzly o vzdálenosti dvou skoků, vybere se jako MPR ten, který dokáže pokrýt více takových uzlů (obr. 3.3).
3. Definování informace pro MPR (MPR Declaration): Jakmile má každý uzel aktualizovaný seznam všech jedno a dvou-skokových uzlů včetně MPR uzlů, vysílá v pravidelných intervalech TC zprávy, které obsahují sekvenční číslo a seznam všech svých MPR uzlů. Zprávy typu TC jsou zasílány pouze uzlům MPR, které je dále šíří do sítě přes ostatní MPR uzly. Kontrolní zprávy se



Obr. 3.2: Neighbor list uzlu S



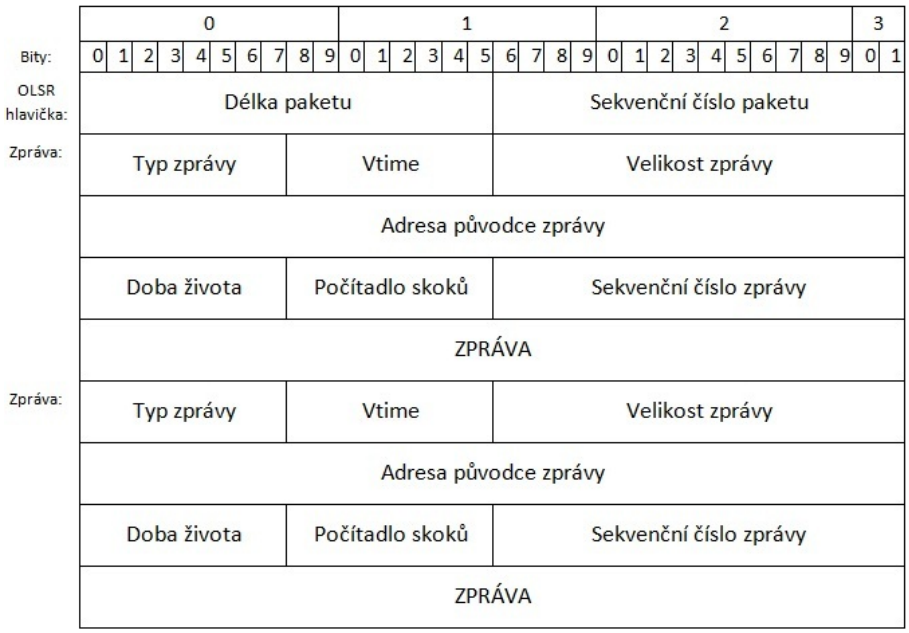
Obr. 3.3: Nastavení MPR uzlů

tak šíří pouze přes vybrané uzly a nezahlcují celé pole. Díky TC zprávám jsou vytvořeny cesty ke všem uzlům v síti.

4. Sestavení směrovacích tabulek (Routing Table Construction): Každý uzel si na základě přijatých TC zpráv vytváří tabulku topologie (Topology Table). Na základě tabulek topologie jsou pak počítány směrovací tabulky. Směrovací tabulky obsahují následující informace: cíl trasy, další skok a počet skoků a jsou aktualizovány při každá změně tabulky topologie nebo seznamu sousedů [15].

3.2 OLSR paket

Základní uspořádání OLSR paketu je zobrazeno na obr. 3.4 (s vynecháním IP a UDP hlavičky) [13].



Obr. 3.4: Struktura OLSR paketu

Hlavička OLSR paketu je tvořena polem délka paketu a sekvenční číslo paketu.

- Délka paketu (Packet Length) - určuje délku paketu v bajtech.
- Sekvenční číslo paketu (Packet Sequence Number) - označuje číslo paketu, které je zvětšeno o jedna pokaždé, když je přenášen nový OLSR paket. Samostatné pořadové číslo je definováno pro každé rozhraní uzlu tak, aby bylo možné jednoznačně identifikovat přenesená data.

IP adresu rozhraní, přes které byl předán paket, je možné získat z hlavičky IP paketu. Pokud paket neobsahuje žádné zprávy (tzn. délka paketu je menší nebo rovna

velikosti hlavičky paketu), musí být paket zahozen. Pro IPv4 adresy to znamená, že pakety, kde délka paketů je menší než 16, musí být zahozeny.

Většina OLSR paketů je složena z jedné nebo více OLSR zpráv. Tyto zprávy mají pevně daný formát, který obsahuje následující položky [13].

- Typ zprávy (Message Type) - udává, jaký typ zprávy se nachází v poli Message. Zprávy v rozsahu 0-127 jsou rezervovány pro OLSR a zprávy o rozsahu 128-255 slouží pro uživatelská rozšíření protokolu.
- Vtime pole - označuje, jak dlouhou dobu po přijetí má uzel brát v úvahu informace obsažené ve zprávě za platné, pokud nejsou novější aktualizace informací k dispozici. Doba platnosti je reprezentována mantisou (čtyři nejvyšší bity Vtime pole) a jejím exponentem (čtyři nejnižší bity Vtime pole).
- Velikost zprávy (Message Size) - udává celkovou velikost zprávy, která se udává v bajtech. Velikost se počítá od pole Typ zprávy až po začátek následujícího pole Typ zprávy nebo pokud neexistuje další pole Typ zprávy, měří se až do konce paketu.
- Adresa původce zprávy (Originator Address) - obsahuje adresu uzlu, který vytvořil tuto zprávu. Toto pole by se nemělo zaměňovat se zdrojovou adresou z IP hlavičky, která se mění pokaždé, kdy je tato zpráva přeposlána.
- Doba života (Time To Live, TTL) - určuje maximální možný počet skoků mezi uzly. Před každým přeposláním zprávy musí být tato hodnota snížena o jedna. Pokud uzel obdrží zprávu s TTL rovnou 0 nebo 1, nesmí ji už za žádných okolností přeposlat dál. Za normálních okolností nenastane situace, kdy uzel obdrží zprávu s TTL rovno 0. Správným nastavením tohoto pole je možné kontrolovat dosah doručení zpráv.
- Počítadlo skoků (Hop Count) - udává počet již vykonaných skoků. Před každým přeposláním zprávy musí být tato hodnota zvýšena o jedna. U původce zprávy je tato hodnota nastavena na 0.
- Sekvenční číslo zprávy (Message Sequence Number) - označuje pořadové číslo zprávy, které se zvětšuje o jedna pro každou zprávu přenášenou od daného uzlu.
- Zpráva (Message) - obsahuje samotnou zprávu OLSR paketu (HELLO zpráva, TC zpráva, MID zpráva, ...).

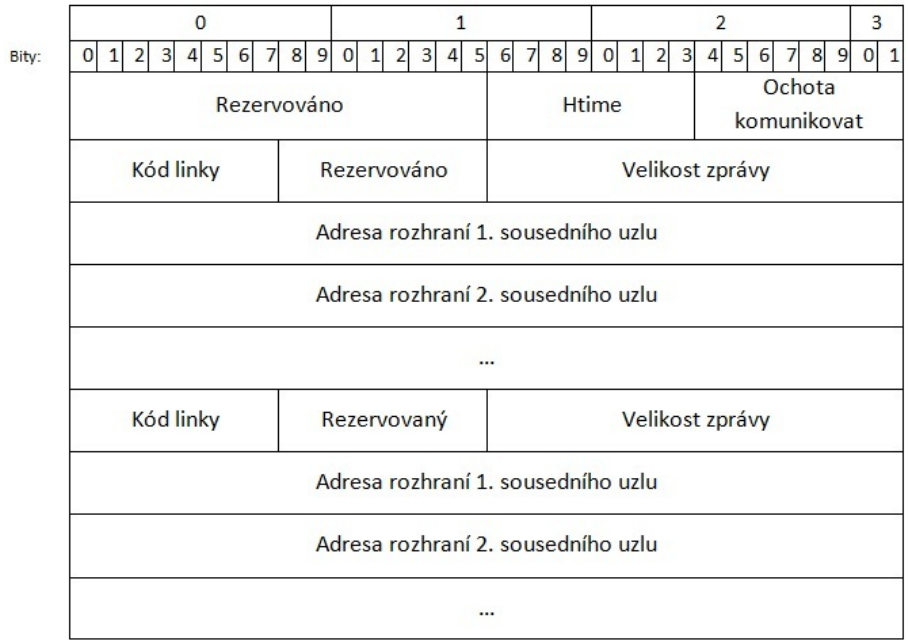
3.2.1 Řídící zprávy v OLSR

HELLO zpráva

HELLO zprávy jsou vysílány všem přímým sousedům uzlu. Tyto zprávy se používají pro zjišťování sousedů (Neighbor discovery) a pro výpočet MPR uzlů. V HELLO

zprávě uzel posílá informace o všech známých linkách a uzlech. HELLO zpráva je zobrazena na obr. 3.5 a obsahuje následující pole:

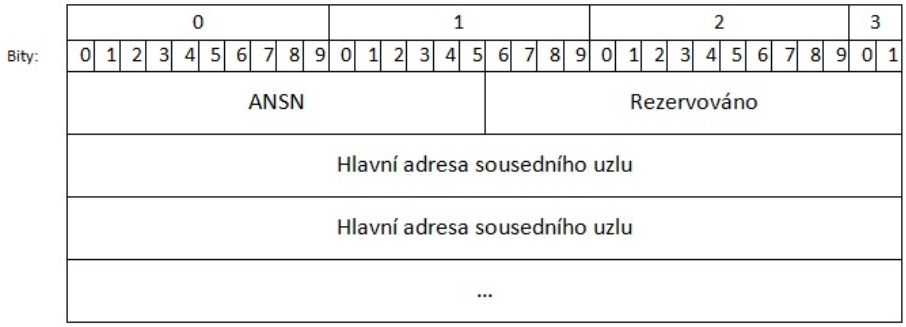
- Rezervováno (Reserved) - toto pole musí být nastaveno na „0000000000000“.
- Htime pole - určuje platnost jednotlivých HELLO zpráv.
- Ochota komunikovat (Willingness) - udává ochotu přenášet data pro ostatní uzly. Ochota uzlu může být nastavena na jakékoli celé číslo od 0 do 7. Existuje několik typů ochoty: ochota WILL_NEVER (odpovídá číslu 0, uzel není ochoten přenášet data) nesmí být nikdy použita pro MPR, ochota WILL_ALWAYS (číslo 7, ochoten vždy přenášet data) je ideální pro MPR a ochota WILL_DEFAULT (číslo 3). Uzly mají ve výchozím nastavení přiřazenou ochotu WILL_DEFAULT. Uzel může dynamicky měnit ochotu komunikovat podle toho, jak se jeho situace mění. Ochota komunikovat může být závislá také na kapacitě baterie.
- Kód linky (Link Code) - obsahuje informace o spojení mezi rozhraním odesílatele a následujícím sousedem. Také specifikuje informace o stavu souseda (např.: jestli se jedná o MPR).
- Velikost zprávy (Link Message Size) - udává celkovou velikost zprávy, která je v bajtech. Velikost se počítá od pole Kód linky až po začátek následujícího pole Kód linky nebo pokud neexistuje další pole Kód linky, měří se až do konce paketu.
- Adresa rozhraní sousedního uzlu (Neighbor Interface Address) - obsahuje adresu rozhraní sousedních uzlů do vzdálenosti dvou skoků (TTL = 2) [13].



Obr. 3.5: Struktura HELLO zprávy v OLSR

Topology Control (TC) zpráva

TC zprávy informují o stavu jednotlivých tras. Tyto zprávy se využívají k optimalizování přenosu informací a výběru MPR. Formát TC zprávy je následující (obr. 3.6) a obsahuje tyto pole:



Obr. 3.6: Struktura TC zprávy v OLSR

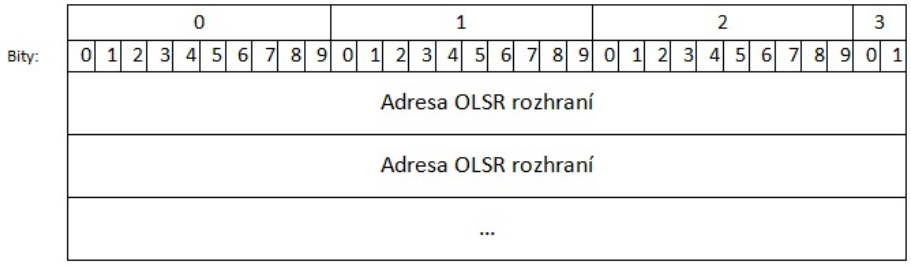
- ANSN (Advertised Neighbor Sequence Number) - obsahuje pořadové číslo informací v TC zprávě. Pokaždé, když uzel detekuje změny ve svém okolí, změní se obsah zasílané TC zprávy a tím se zvětší hodnota pořadového čísla. Uzel vždy vybírá TC zprávy s největší hodnotou pole ANSN. Tím je zaručena „aktuálnost“ informací.
- Rezervováno (Reserved) - toto pole musí být nastaveno stejně jako u HELLO zprávy na hodnotu „0000000000000“.
- Hlavní adresa sousedního uzlu (Advertised Neighbor Main Address) - obsahuje adresu sousedního uzlu. V TC zprávě jsou uvedeny všechny hlavní adresy sousedních uzlů. Pokud je kapacita TC zprávy vyčerpána a všechny adresy sousedních uzlů v ní nejsou uvedeny, musí se vygenerovat více TC zpráv, aby byly uvedeny všechny adresy [13].

Multiple Interface Declaration (MID) zpráva

MID zprávy se používají k rozšíření informací o uzlech, které mají více než jedno komunikačních rozhraní. Struktura MID zprávy je uvedena na obr. 3.7 a obsahuje adresy všech OLSR rozhraní (OLSR Interface Address), které uzel používá ke komunikaci [13].

3.3 Kvalita služeb (QoS) v OLSR

Rozšířením protokolu OLSR o podporu Quality of Service (dále jen QoS) vznikl nový směrovací protokol QOLSR (QoS Optimized Link State Routing). Pro potřeby QoS



Obr. 3.7: Struktura MID zprávy v OLSR

byly rozšířeny HELLO a TC zprávy. Žádné další řídící zprávy nejsou vytvářeny. V QOLSR uzel měří QoS parametry (zpoždění, jitter - kolísání zpoždění, dostupnou šířku pásma a další) na spojeních ke svým sousedům. Tyto informace jsou používány k počítání QoS MPR (QMPR) a směrovacích tabulek [3].

3.4 Výhody OLSR

- Minimální zpoždění při vzniku požadavku na komunikaci,
- minimální kolísání zpoždění při přenosu datových informací,
- vhodný pro použití ve velkých sítích s velkou hustotou,
- využitím MPR dosahuje větší efektivity oproti klasickým LS protokolům,
- snadno rozšiřitelný o QoS - QOLSR.

3.5 Nevýhody OLSR

- Vysoká výpočetní náročnost v jednotlivých uzlech,
- velký kontrolní provoz (režie),
- složitější implementace,
- ve velmi řídkých sítí může docházet k tomu, že každý soused uzlu je MPR,
- pro přenos informací jsou používány pouze trasy s MPR uzly, což odstraňuje část linek, které by mohly sloužit jako záložní (OLSR snižuje redundanci v síti).

4 AD HOC ON DEMAND DISTANCE VECTOR ROUTING

Ad Hoc On Demand Distance Vector (AODV) je velice populární směrovací protokol v MANET sítích. Jedná se o reaktivní tabulkově řízený směrovací protokol, pro který je typické, že cesta k cíli se sestavuje až v případě požadavku na přenos a přenos paketů je řízen informacemi ze směrovacích tabulek všech uzlů na cestě přenosu. To zmenšuje směrovací režii, ale zavádí větší zpoždění při požadovaném nastavení trasy. AODV vyvinuly společně dvě univerzity v Nokia Research Centru. Byla to Univerzita Kalifornie v Santa Barbaře a Univerzita Cincinnati pod vedením C. Perkinsa, E. Belding-Royera a S. Dase. Specifikace je dostupná v RFC 3561 [12].

Tento protokol používá k nalezení cest jednoduchý mechanismus dotaz-odpověď (request-reply) a také využívá HELLO zprávy pro nalezení sousedů. Sekvenční čísla slouží jako identifikátor „aktuálnosti cesty“, takže jsou použity jen nejaktuálnější směrovací informace.

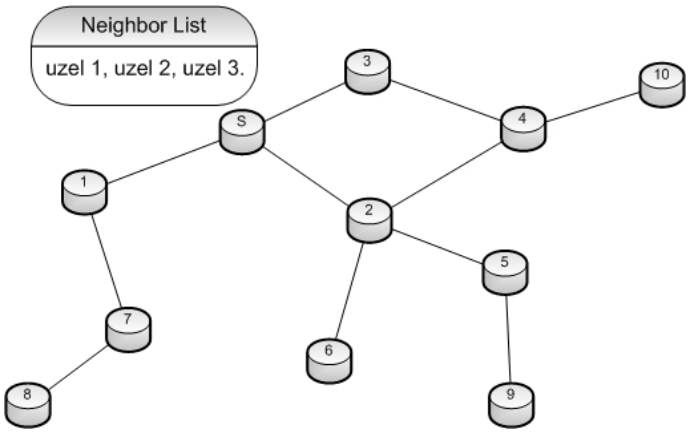
V síti, kde je používán směrovací protokol AODV, má každý paket přidělené tzv. časové razítko (time-stemp). Pokud se paket pohybuje v síti delší dobu, než dovoluje časové razítko, dojde k zahození paketu. Tímto se předchází zacyklení [2].

4.1 Princip protokolu AODV

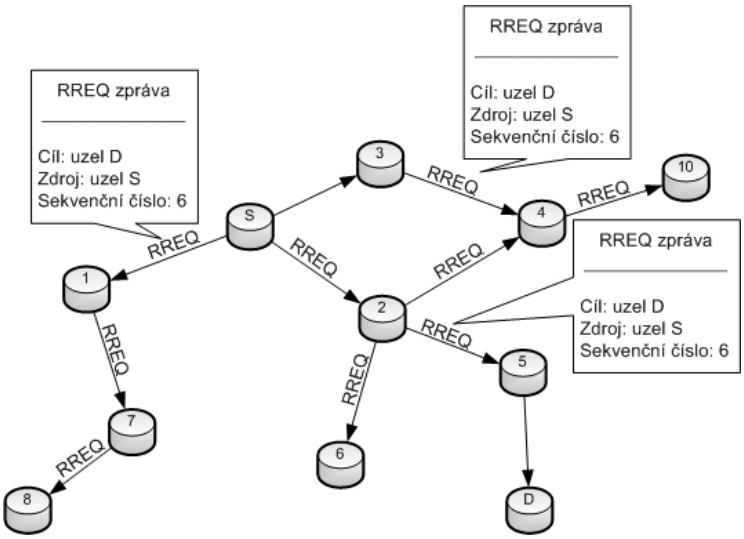
Princip protokolu AODV je možné rozdělit do následujících tří kroků, které popisují jeho činnost:

1. Nalezení cesty od zdroje k cíli: Proces směrování je zahájen uzlem S, který potřebuje poslat data uzlu D, pro který nemá uložené žádné informace ve své směrovací tabulce. Díky pravidelnému zasílání HELLO zpráv (TTL = 1), má uzel S v paměti uloženy ID (IDentification) svých sousedů. Okolní uzly, které jsou v jeho dosahu, si po přijetí této HELLO zprávy aktualizují svoji tabulku sousedů (Neihbors list) (viz. obr. 4.1). Uzel S tedy vyšle požadavek všem svým sousedům o spojení s uzlem D (obr. 4.2). Tento požadavek je definován zprávou RREQ (Route Request). Pokud žádný ze sousedních uzlů, který přijal RREQ, nezná cestu k uzlu D nebo nemá uzel D ve svém rádiovém dosahu, posílá zprávu RREQ dál svým sousedům. Jakmile uzel předá paket s žádostí, nastaví zpáteční cestu od sebe k uzlu S tak, že zaznamená adresu sousedního uzlu, od kterého přijal první kopii RREQ zprávy. Podobně, když je RREQ kontrolní zpráva předána směrem k cíli, uzel automaticky nastaví zpáteční cestu ze všech uzlů zpátky ke zdroji. Další zpáteční cesty jsou smazány po uplynutí časového intervalu. Sekvenční číslo ve zprávě slouží proti vytváření smyček. Toto se

opakuje, dokud se nenajde uzel, který zná cestu k uzlu D (informaci o uzlu D má uloženou v Neighbors listu). Poté je zpět posíláno potvrzení cesty RREP (Route Reply) s počtem skoků (Hop Count) (viz. obr. 4.3). Potvrzení RREQ je zasláno všem sousedním uzlům, aby si do své směrovací tabulky (Routing List) přidali cestu k dalšímu uzlu.

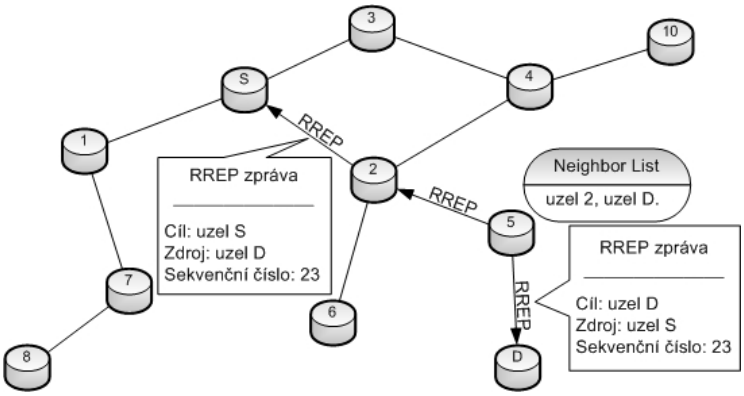


Obr. 4.1: Neighbor list pro uzel S

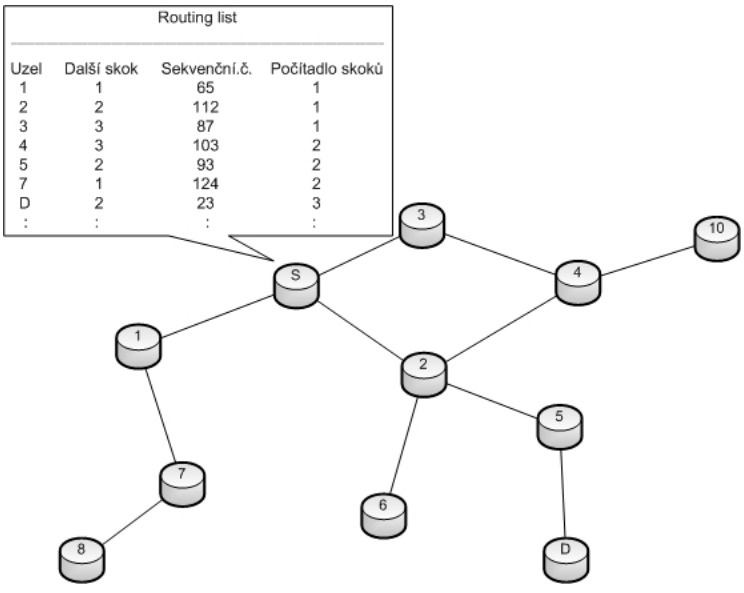


Obr. 4.2: Vysílání požadavku o spojení - RREQ zpráva

2. Zaznamenání cest na všech mezilehlých uzlech patřících do vytvářené cesty: Na konci fáze objevu jsou v důsledku relace paketů žádosti-odpovědi vytvořeny nové směrovací stavy v jednotlivých uzlech (obr. 4.4). Každý uzel si udržuje směrovací tabulku, která je definována koncovým uzlem a jedním ze sousedních uzlů, který je mezilehlým prvkem na cestě. Cesta dopředu je zrušena, pokud není používána v průběhu určeného časového intervalu vypršení cesty. Pokaždé, kdy je cesta používána, je časový interval vypršení cesty resetován.

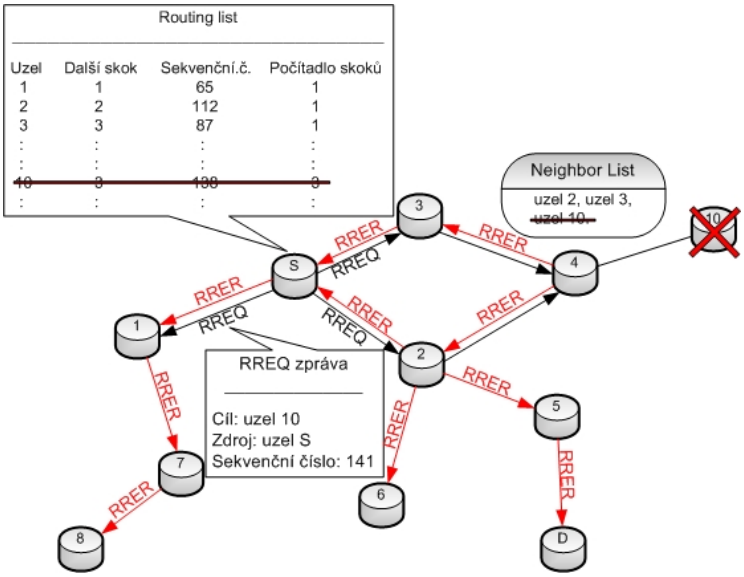


Obr. 4.3: Poslání potvrzení trasy - RREP zpráva



Obr. 4.4: Zaznamenání směrovacích informací na uzlu S

3. Udržování aktuálních cest: Trasa je udržována na základě pravidelného zasílání HELLO zpráv. Při zjištění poruchy spojení, uzel odešle zprávu RERR (Route Error) všem svým aktivním předřazeným sousedům, aby zrušili všechny trasy, které používají právě toto přerušené spojení. Tyto uzly postupně přenášejí tento paket svým předřazeným sousedům, a tak se nakonec všechny aktivní zdroje dozvědí o této poruše (obr. 4.5). Po obdržení zprávy RERR si zdroj vyžádá další žádost o směrování [15].



Obr. 4.5: Zjištění poruchy spojení - uzel 10 se vzdálil od uzlu 4

4.2 AODV zprávy

V této kapitole budou popsány jednotlivé zprávy, které jsou posílány při směrování v MANET sítích využívajících směrovací protokol AODV.

4.2.1 Route Request (RREQ) zpráva

RREQ zprávy jsou posílány všem přímým sousedům uzlu jako požadavky o spojení s cílovým uzlem. Formát RREQ zprávy je zobrazen na obr. 4.6 a význam jednotlivých polích je:

- Typ zprávy (Type) - je nastaven na hodnotu 1.
- J - příznak pro připojení (Join). Rezervováno pro multicast.
- R - příznak pro opravu (Repair). Rezervováno pro multicast.

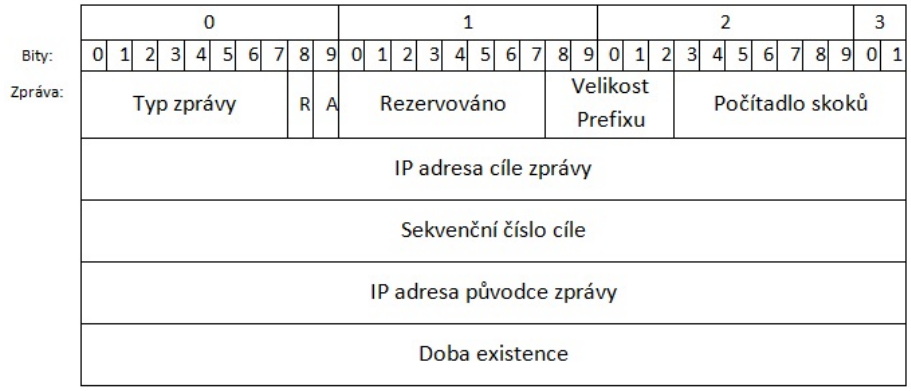
Obr. 4.6: Struktura RREQ zprávy v AODV

- G - příznak nadbytečnosti (Gratuitous) RREP zprávy. Uvádí, zda by měla být RREP zpráva odeslána také uzlu, který je uveden jako cíl zprávy RREQ v poli IP adresa cíle zprávy. Tato RREP zpráva se označuje jako „nadbytečná“.
- D - příznak cíle (Destination), který uvádí, jestli může cíl reagovat na tuto RREQ zprávu.
- U - příznak neznámého (Unknown) pořadového čísla. Udává se, když cílové pořadové číslo není známo.
- Rezervováno (Reserved) - toto pole je nastaveno na hodnotu 0. To znamená, že je ignorováno při přijetí.
- Počítadlo skoků (Hop Count) - obsahuje počet skoků od původce zprávy k uzlu, který vyřizuje žádost.
- PREQ ID - obsahuje pořadové číslo, které jednoznačně identifikuje konkrétní RREQ zprávu, pokud je používáno ve spojení s IP adresou původce zprávy.
- IP adresa cíle zprávy (Destination IP Address) - udává IP adresu místa určení.
- Sekvenční číslo cíle (Destination Sequence Number) - nese poslední pořadové číslo přijaté původcem zprávy pro jakoukoli cestu směrem k cíli.
- IP adresa původce zprávy (Originator IP Address) - určuje IP adresu uzlu, který vytvořil požadavek na směrování.
- Sekvenční číslo původce (Originator Sequence Number) - obsahuje aktuální pořadové číslo, které je použito na vstupu trasy směřující k původci žádosti [12].

4.2.2 Route Reply (RREP) zpráva

RREP zprávy slouží pro potvrzení cesty k uzlu a jsou odesílány jako reakce na RREQ zprávy. Složení RREP zprávy je uvedeno na obr. 4.7 a obsahuje tyto jednotlivé pole:

- Typ zprávy (Type) - je nastaven na hodnotu 2.
- R - příznak pro opravu (Repair). Rezervováno pro multicast.
- A - příznak pro požadování potvrzení (Acknowledgment).
- Rezervováno (Reserved) - toto pole je nastaveno na hodnotu 0. Ignorováno při přijetí.
- Velikost Prefixu (Prefix Size) - pokud je toto 5-bitové pole nenulové, tak určuje, jestli může být uvedený další skok použit pro více uzlů se stejným směrovacím prefixem jako požadované místo určení.
- Počítadlo skoků (Hop Count) - obsahuje počet skoků od původce zprávy k cíli.
- IP adresa cíle zprávy (Destination IP Address) - určuje IP adresu místa určení.
- Sekvenční číslo cíle (Destination Sequence Number) - nese pořadové číslo cíle spojené s cestou.
- IP adresa původce zprávy (Originator IP Address) - obsahuje IP dresu uzlu, ze kterého byla RREP zpráva odeslána jako reakce na RREQ zprávu.
- Doba existence (Lifetime) - udává čas v milisekundách, po který uzly, které obdrželi RREP zprávu, považují cestu za platnou [12].



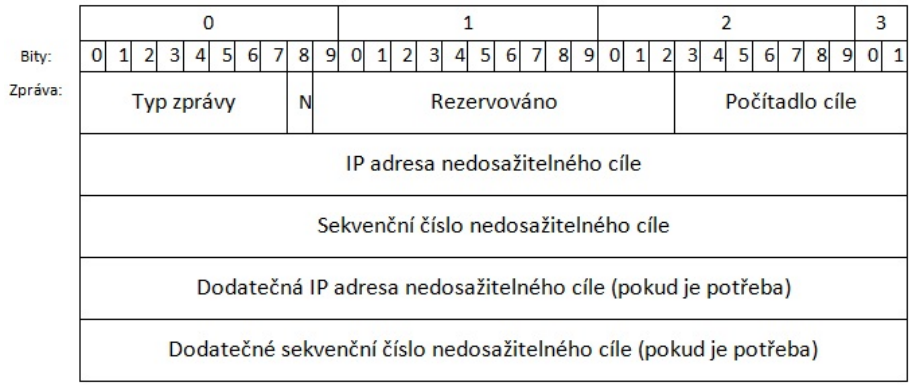
Obr. 4.7: Struktura RREP zprávy v AODV

4.2.3 Route Error (RERR) zpráva

RERR zprávy jsou odesílány při zjištění poruchy spojení všem svým sousedům. Formát Route Error (RERR) zprávy je uveden na obr. 4.8 a jeho pole jsou:

- Typ zprávy (Type) - je nastaven na hodnotu 3.
- N - příznak pro neodstranění (No delete) trasy a nastavení uzlu, aby prováděl lokální opravu linky. Při výpadku linky může uzel provést lokální opravu linky tak, že odešle RREQ zprávu pro vybudování nové části cesty k cíli. Uzel provádějící lokální opravu musí ukládat pakety během doby, po kterou čeká na odpověď od nějaké trasy.

- Rezervováno (Reserved) - toto pole je nastaven na hodnotu 0. Ignorováno při přijetí.
- Počítadlo cíle (DestCount) - nese počet nedostupných destinací. Hodnota musí být alespoň 1.
- IP adresa nedosažitelného cíle (Unreachable Destination IP Address) - obsahuje IP adresu nedostupného uzlu (např.: z důvodu přerušení spojení).
- Sekvenční číslo nedosažitelného cíle (Unreachable Destination Sequence Number) - udává pořadové číslo pro nedosažitelný uzel [12].

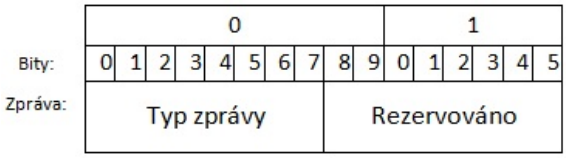


Obr. 4.8: Struktura RERR zprávy v AODV

4.2.4 Route Reply Acknowledgment (RREP-ACK) zpráva

RREP-ACK zpráva musí být odeslána jako odpověď na zprávu RREP, která má nastavený příznak pro požadování potvrzení (Acknowledgment). Toto obvykle nastane, pokud hrozí nebezpečí, že jednosměrné spojení brání dokončení cyklu objevení trasy. Zpráva RREP-ACK je zobrazena na obr. 4.9 a význam jednotlivých polí je:

- Typ zprávy (Type) - je nastaven na hodnotu 4.
- Rezervováno (Reserved) - pole nastaveno na hodnotu 0. Ignorováno při přijetí [12].



Obr. 4.9: Struktura RREP-ACK zprávy v AODV

4.3 Kvalita služeb (QoS) v AODV

Hlavní myšlenka QoS v AODV spočívá v rozšíření RREQ a RREP zpráv během hledání cesty. Mezi položky, které rozšiřují jednotlivé zprávy a směrovací tabulky, patří maximální zpoždění, minimální dostupná šířka pásma, seznam zdrojů vyžadujících záruky zpoždění a seznam zdrojů vyžadujících garanci šířky pásma. Uzel, který přijímá RREQ zprávy s rozšířením o QoS, musí být schopen splnit požadavky k tomu, aby buď přeposlal RREQ zprávu anebo poslal RREP zprávu ke zdroji. Pokud po sestavení spojení nějaký uzel zjistí, že již nemohou být požadované QoS parametry udržovány, vytvoří zprávu QOS_LOST a odešle ji ke zdroji. Poté je na zdroji, aby zařídil lepší trasu, kde budou dodrženy požadované QoS parametry [3].

4.4 Výhody AODV

- Oproti klasickým směrovacím protokolům (DV a LS protokoly) redukuje počet režijních zpráv,
- optimální pro použití v řídkých a středně hustých sítích,
- po stanovení spojení k cíli, neobsahují již datové pakety žádné služební informace potřebné pro směrování,
- pomocí sekvenčních čísel je zaručeno, že směrovací tabulky obsahují pouze nejaktuálnější informace o síti.

4.5 Nevýhody AODV

- Není vhodný pro použití ve velkých sítích s velkou hustotou uzlů,
- větší zpoždění při vzniku požadavku na komunikaci,
- složitější implementace QoS.

5 OPNET MODELER

Program OPNET Modeler (dále jen OM) je simulační prostředí, které slouží pro návrh, simulaci a analýzu odlišných síťových technologií. Tento program byl vyvinut firmou OPNET Technologies Inc. OM je hierarchicky a objektově orientován. Grafické prostředí odráží reálné rozmístění jednotlivých síťových prvků a na nejnižší úrovni je chování jednotlivých komponent zapsáno v jazyce C/C++. Dále obsahuje široké možnosti v oblasti simulace a analýzy výsledků. Této vlastnosti se využívá hlavně při ověření chování reálných objektů v různých podmínkách.

Výsledky simulací můžeme vygenerovat do souborů ve formátech XML, HTML nebo data uložit jako tabulku. OM obsahuje také prohlížeč animací, díky kterému je možné sledovat průběh simulace. Simulace v programu OM jsou oproti reálným hodnotám zrychlené, takže je možné nasimulovat několika hodinové chování sítě během pár minut.

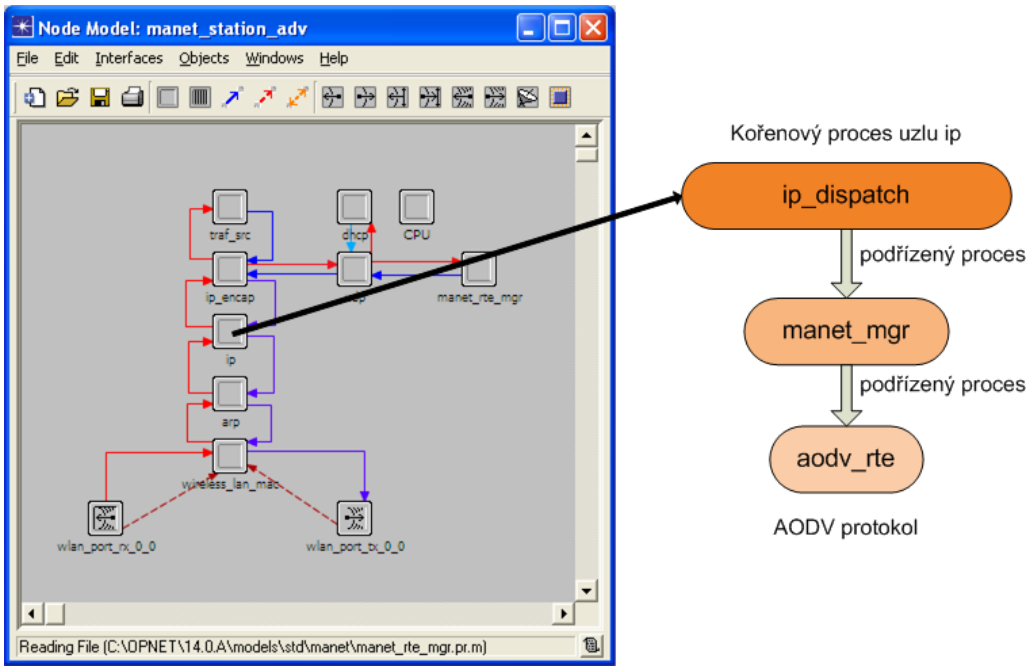
Velkou výhodou OM je, že všechny dostupné knihovny (např. síťových prvků, definic paketů, linkových vrstev apod.) mají dostupný zdrojový kód, takže ho lze upravovat, což je velice výhodné, je-li potřeba např. vyzkoušet nové možnosti nějakého standardního prvku [4].

5.1 MANET směrovací protokoly v prostředí OM

Program OM poskytuje několik modelů MANET směrovacích protokolů, které jsou sjednoceny s modely IP a bezdrátových sítí LAN. Navíc je struktura MANET sítě dostupná pro rychlý rozvoj nových modelů MANET protokolů. Firma OPNET vyvíjela model MANET sítě v úzké spolupráci s více než 150 odborníky na problematiku MANET sítí z různých oblastí (vláda, průmysl, akademická obec a dalších). V prostředí OM je možné simulovat tyto směrovací protokoly MANET sítí: AODV, DSR, OLSR, OSPFv3, TORA a GRP [9].

5.2 Implementace AODV protokolu v prostředí OM

AODV protokol je v OM implementován na síťové vrstvě. Jak je patrné z obr. 5.1, je ip_dispatch kořenový proces IP protokolu a jeho podřízený proces (child process) je manet_mgr. Manet_mgr vystupuje jako manažer procesu pro AODV a poskytuje společné rozhraní k více MANET směrovacím protokolům (AODV, DSR, GRP a TORA). Manet_mgr je zodpovědný za vznik AODV procesu, pokud je na uzlu nastaven AODV protokol.



Obr. 5.1: Implementace AODV protokolu v prostředí OM

V prostředí OM jsou používány následující typy AODV paketů: AodvT_Rreq, AodvT_Rrep a AodvT_Rerr. Složení těchto paketů odpovídá AODV zprávám, které jsou posílány v MANET sítích při směrování (viz. kapitola 4.2).

Model AODV protokolu obsahuje následující dva časovače:

- Route Entry Expiry - každá cesta v AODV směrovací tabulce je spojena s časovačem platnosti trasy. Pokud trasa není použita do konce uplynutí této doby, je tato trasa označena za neplatnou a nemůže být použita pro přenos paketů. Trasa je odstraněna po vypršení DELETE_TIME sekund. Hodnota časovače se resetuje pokaždé, kdy je cesta používána.
- Route Request Expiry - AODV spolupracuje s časovačem nalezení trasy pro každou odeslanou žádost pro nalezení trasy. Když nedorazí odpověď do vypršení časovače a není dosažen limit opakování žádosti, je odeslána další žádost. Pokud je již limit dosažen, jsou všechny pakety pro tuto destinaci zahozeny [10].

5.2.1 Konečný stavový automat - ip_dispatch

Modul ip v modelu uzlu manet_station_adv vychází z procesního modelu ip_dispatch (viz. obr. 5.2), který plní základní funkce síťové vrstvy. IP pakety jsou směrovány k cíli na základě IP adresy.

Procesní model ip_dispatch vytváří a spolupracuje s velkou řadou podřízených procesů. Směruje příchozí pakety z/do podřízených procesů a do vyšších vrstev. Ke

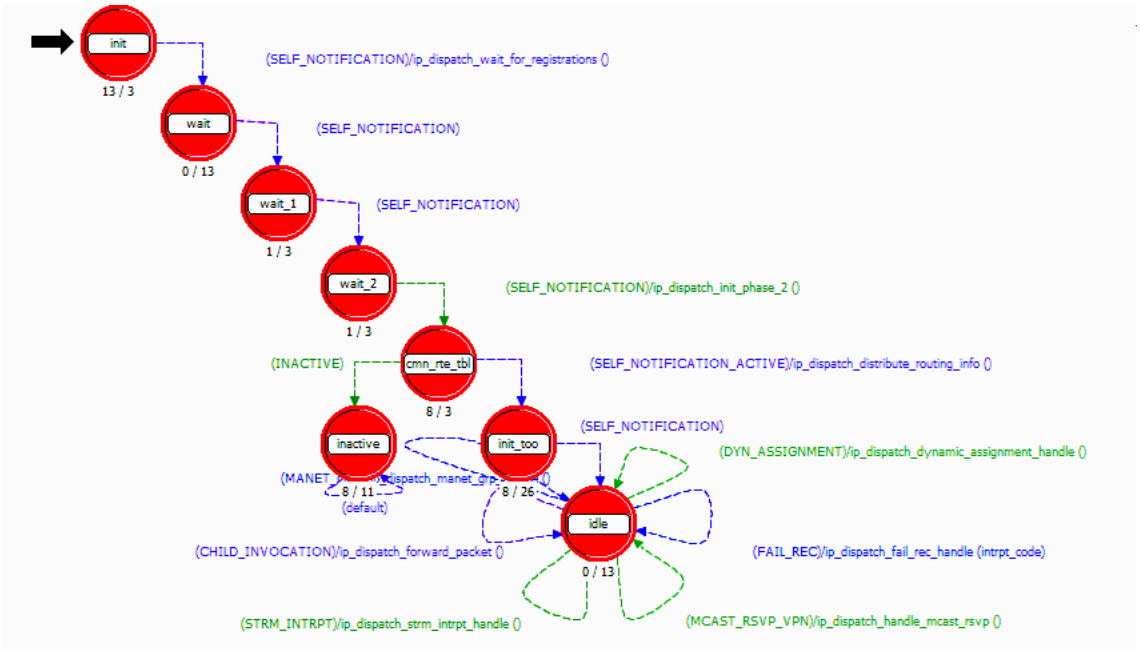
směrování používá model ip_dispatch následující podřízené procesy:

- ip_rte_central_cpu,
- ip_rte_cloud,
- ip_rte_distrib_cpu,
- ip_rte_slot.

Ve výchozím stavu *init* tohoto konečného stavového automatu ip_dispatch (obr. 5.2) probíhá inicializace modelu a stavových proměnných. Ve stavech *wait*, *wait_1* a *wait_2* je získán typ příchozího přerušení a podle získaného typu přejde proces do následujícího stavu. Stav *cmn_rte_tbl* se používá k zahájení počátečního přerozdělování směrovacích informací mezi směrovací protokoly nakonfigurovaných na tomto uzlu. To se provádí pomocí volání funkce `ip_cmn_rte_table_redistribute ()` při přechodu z tohoto stavu.

Dalším stavem tohoto procesního modelu je stav *inactive*. Do tohoto stavu je vstoupeno, pokud má uzel vypnuty všechny rozhraní, tedy inicializace neproběhla správně. Pokud vše proběhlo správně, je dokončena inicializace. To je provedeno ve stavu *init_too*. Tento stav poté vytvoří a spustí vhodnou skupinu podřízených procesů.

Poslední stav je *idle*, kde proces čeká na příchozí přerušení. Podle typu příchozího přerušení je vykonána jedna z mnoha funkcí síťové vrstvy (např. přenos směrovacích informací, směrování přijatých datových jednotek z podřízených procesů do jiných procesů popřípadě modulů a další) [5].

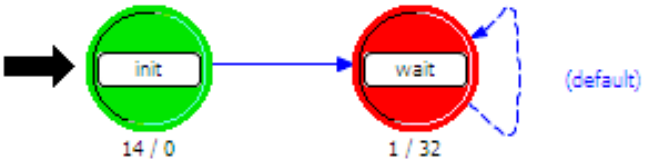


Obr. 5.2: Konečný stavový automat - ip_dispatch

5.2.2 Konečný stavový automat - manet_mgr

Manet_mgr je hlavní plánovací proces, který vytvoří příslušný ad-hoc směrovací protokol běžící na uzlu. Je umístěn v IP modulu jako podřízený proces procesu ip_dispatch.

Výchozím stavem konečného stavového automatu manet_mgr (obr. 5.3) je stav *init*, kde pomocí funkce `manet_mgr_sv_init ()` probíhá inicializace stavových proměnných, funkce `manet_mgr_routing_protocol_determine ()` definuje MANET protokol běžící na všech rozhraních a vytvoření a uplatnění příslušného procesu pro MANET směrování pomocí funkce `manet_mgr_routing_process_create ()`. Tato funkce vytvoří a vyvolá příslušný podřízený procesní model směrovacího protokolu (v našem případě aodv_rte). Dalším stavem tohoto automatu je stav *wait*. Zde se čeká na volání od jiných procesů. Pokud je obdrženo volání, je zkontrolováno, zda byl proces volán od potomka nebo od nějakého procesoru.



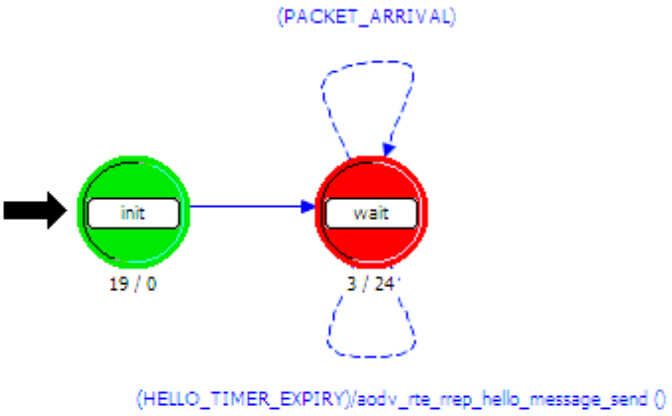
Obr. 5.3: Konečný stavový automat - manet_mgr

5.2.3 Konečný stavový automat - aodv_rte

Proces aodv_rte (obr. 5.4) směrovacího protokolu AODV je podřízený proces manet_mgr procesu. Pokud je AODV protokol nakonfigurován na uzlu, aodv_rte proces zapříčiní to, že proces manet_mgr spustí AODV protokol.

Tento procesní model obsahuje dva stavy: *init* a *wait*. Ve stavu *init* probíhá inicializace stavových proměnných příkazem `aodv_rte_sv_init ()`, registrace místních statistik pomocí `aodv_rte_local_stats_reg ()`, analýza atributů a vytvoření vyrovnávacích pamětí funkcí `aodv_rte_attributes_parse_buffers_create ()`, registrace AODV směrovacího procesu jako vyšší vrstva v IP protokolu příkazem `Ip_Higher_Layer_Protocol_Register("aodv", &higher_layer_proto_id)` a přidání cesty k sousednímu uzlu pouze tehdy, pokud to není brána MANET sítě. Dále následuje stav *wait*, který čeká na příchod paketu a také volá funkci pro odeslání

HELLO zprávy v případě vypršení časovače HELLO_TIMER. Dále je zde prováděna aktualizace životnosti trasy, předchozího skoku a dalšího skoku k určení směrování paketů.



Obr. 5.4: Konečný stavový automat - aodv_rte

5.2.4 Blok kódů procesního modelu aodv_rte

V této kapitole budou popsány důležité bloky zdrojového kódu procesního modelu aodv_rte. První blokem jsou stavové proměnné (SV). Zde jsou definovány proměnné, které si zachovávají svou hodnotu v celém procesním modelu. Dalším důležitým blokem je hlavičkový blok (HB), ve kterém jsou definovány všechny hlavičkové soubory, které tento model používá, a funkční prototypy pro funkce, které se vyskytují ve funkčním bloku (FB).

Nejdůležitějším blokem procesního modelu aodv_rte je funkční blok (FB). Zde jsou uvedeny funkce, které zajišťují správnou činnost AODV směrovacího protokolu:

- `aodv_rte_sv_init ()` - zde probíhá inicializace stavových proměnných, přístup do datové paměti, načtení jednotlivých ID, přiřazení jedinečného názvu pro potřeby směrování a vytvoření masky podsítě.
- `aodv_rte_local_stats_reg ()` - slouží pro inicializaci a registraci lokálních statistik.
- `aodv_rte_attributes_parse_buffers_create ()` - načítá jednotlivé atributy (AODV, TTL, ...) a vytváří vyrovnávací paměti funkcí.
- `aodv_rte_add_directly_connected_routes ()` - přidává rozhraní, která jsou přímo připojená k danému uzlu.

- `aodv_rte_pkt_arrival_handle ()` - tato funkce je volána, pokud přišel nějaký paket. Zde probíhá přístup k informacím v příchozím IP datagramu. Nejprve je zkontrolováno, jestli je AODV protokol nastaven na uzlu, který přijal paket. Dále je zkontrolováno, jestli paket nepochází z aplikací vyšších vrstev. Pokud ano, je předán příslušné vrstvě. Nakonec je zjištěno, jestli paket patří mezi AODV zprávy. Pokud ano, je vyčtena hodnota atributu data a options. Z hodnoty `tlv_options->type` je zjištěn typ AODV zprávy (AODVC_ROUTE_REQUEST, AODVC_ROUTE_REPLY, AODVC_HELLO nebo AODVC_ROUTE_ERROR) a poté je zavolána funkce pro zpracování příslušné AODV zprávy.
- `aodv_rte_app_pkt_arrival_handle ()` - tato funkce je volána, pokud dorazí paket, který patří aplikaci z jiné vrstvy. Funkce zajišťuje přeposlání paketu příslušné vrstvě.
- `aodv_rte_rreq_pkt_arrival_handle ()` - do této funkce proces přejde v případě přijetí RREQ zprávy. Zde jsou vyčteny údaje z této zprávy, aktualizovány směrovací tabulky a je aktualizována životnost trasy.
- `aodv_rte_rrep_pkt_arrival_handle ()` - tato funkce je volána, pokud dorazí RREP zpráva. Zde dojde k vyčtení údajů z této zprávy, k aktualizování směrovacích tabulek a aktualizaci životnosti trasy.
- `aodv_rte_rrep_hello_pkt_arrival_handle ()` - funkce, která je volána po přijetí HELLO zprávy. Dochází zde k vyčtení údajů, k aktualizaci směrovacích tabulek a k aktualizaci životnosti trasy pomocí HELLO zprávy. Nakonec je paket zničen.
- `aodv_rte_rerr_pkt_arrival_handle ()` - do této funkce proces přejde v případě přijetí RERR zprávy a dojde k vyčtení údajů z této zprávy.
- `aodv_rte_data_routes_expiry_time_update ()` - tato funkce je volána v případě, že je potřeba aktualizovat životnost trasy.
- `aodv_rte_route_table_entry_update ()` - tato funkce se používá k vytvoření nebo aktualizaci směrovacích tabulek na základě informací obsažených v jednotlivých AODV zprávách.
- `aodv_rte_route_table_entry_from_hello_update ()` - funkce, která slouží k aktualizaci životnosti trasy na základě informací v HELLO zprávě.
- `aodv_rte_route_request_send ()` - tato funkce vytvoří, naplní a odešle RREQ zprávu (rreq_option_ptr).
- `aodv_rte_route_reply_send ()` - funkce, která vytvoří a odešle RREP zprávu (rrep_option_ptr).
- `aodv_rte_grat_route_reply_send ()` - tato funkce vytváří RREP s příznakem nadbytečnosti (viz. kapitola 4.2.1).
- `aodv_rte_ext_route_error_send ()` - tato funkce vytvoří a odešle RERR

zprávu (rerr_option_ptr).

- `aodv_rte_route_error_process ()` - v této funkci se provádí proces zjištění chyby, která zapříčiní odeslání RERR zprávy.
- `aodv_rte_all_pkts_to_dest_send ()` - funkce, která odešle všechny pakety ve frontě na místo určení.
- `aodv_rte_rrep_hello_message_send ()` - tato funkce vytvoří a odešle zprávu HELLO. Pokud vypršel hello interval, je odeslána tato zpráva všesměrově sousedním uzlům. Pro vytvoření HELLO zprávy se používá datová struktura zprávy RREP (rrep_option_ptr). Následně je tato zpráva zapouzdřena do IP datagramu.
- `aodv_rte_entry_expiry_handle ()` - funkce, která zpracovává přerušení, když vyprší životnost trasy. Na základě tohoto přerušení je cesta smazána anebo prohlášena za neplatnou.
- `aodv_rte_rreq_timer_expiry_handle ()` - funkce, která po vypršení časovače nalezení trasy zašle novou RREQ zprávu.
- `aodv_rte_local_repair_attempt ()` - funkce, která provádí místní opravy na základě příznaku pro opravu (viz. kapitola 4.2.3).
- `aodv_rte_local_repair_exists ()` - tato funkce pracuje s příznakem pro opravu. Pokud je příznak nastaven, pak může být místo určení odstraněno ze seznamu destinací.
- `aodv_rte_neighbor_connectivity_table_update ()` - funkce sloužící pro aktualizaci tabulky sousedů.
- `aodv_rte_neighbor_conn_loss_handle ()` - tato funkce se volá, pokud bylo spojení mezi sousedy ztraceno.
- `aodv_rte_ip_datagram_create ()` - tato funkce vytvoří IP datagram, do kterého zapouzdří určitou AODV zprávu.

5.2.5 Další soubory používané v procesním modelu aodv_rte

Jak již bylo zmíněno v předchozí kapitole, procesní model aodv_rte využívá pro svoji činnost hlavičkové soubory. Hlavičkové soubory jsou uloženy ve složce `C:\Program Files\OPNET\16.0.A\models\std\include`. Mezi tyto hlavičkové soubory patří:

- aodv.h - zde jsou definovány konstanty pro přístup do směrovacích tabulek a pro identifikaci časovačů. Dále se zde definují datové struktury pro lokální a globální statistiky, směrovací tabulky, fronty paketů, tabulky žádostí pro nalezení trasy a tabulky sousedů.
- aodv_pkt_support.h - v tomto souboru jsou definovány konstanty pro jednotlivé typy AODV zpráv a pro velikosti těchto zpráv v bitech. Také je zde definována datová struktura pro AODV zprávy (AodvT_Rreq, AodvT_Rrep

a AodvT_Rerr).

- aodv_ptypes.h - zde jsou uvedeny funkční prototypy pro směrovací tabulky, fronty paketů, tabulky žádostí pro nalezení trasy a vytvoření AODV zpráv.

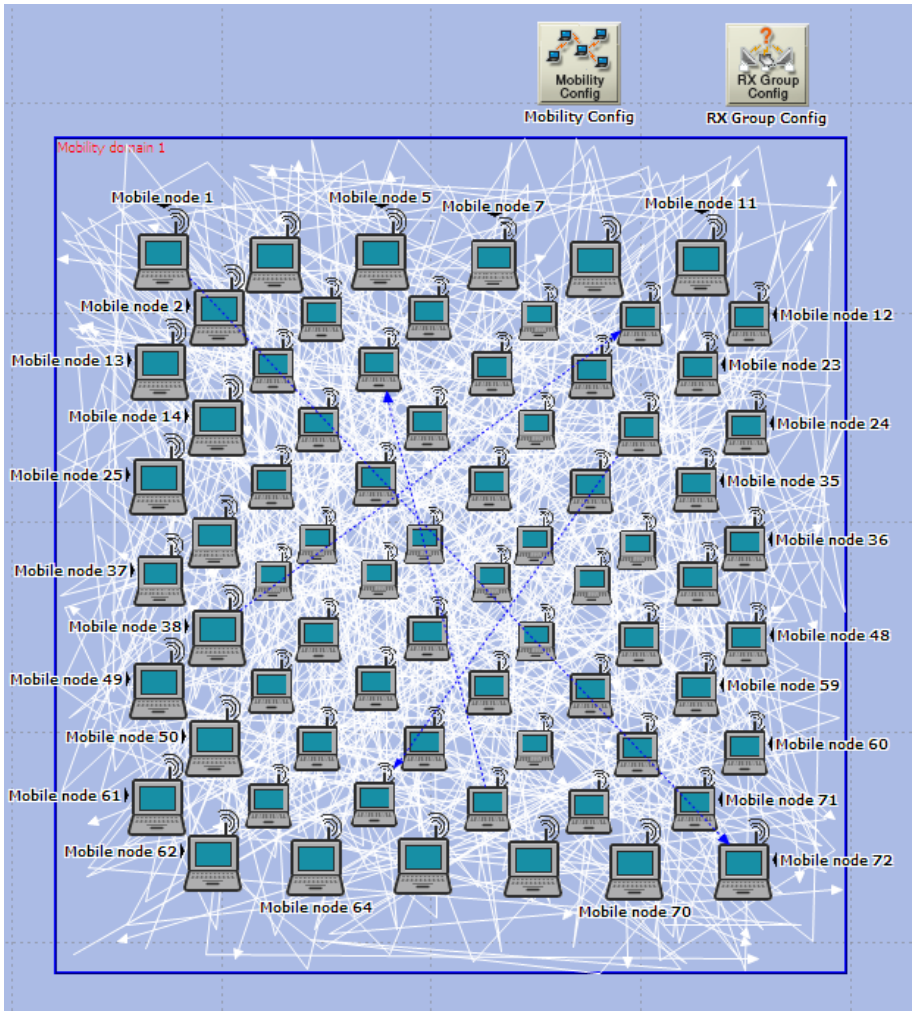
Dále procesní model aodv_rte využívá externí bloky kódů (External C Code). Tyto soubory jsou uloženy ve složce `C:\Program Files\OPNET\16.0.A\models\std\manet`. Mezi tyto bloky kódů patří:

- aodv_packet_queue.ex.c - zde jsou vytvořeny funkce pro práci s frontami paketů uvedené v hlavičkovém souboru aodv_ptypes.h.
- aodv_pkt_support.ex.c - v tomto souboru jsou definovány funkce pro vytvoření AODV zpráv. Funkce `aodv_pkt_support_rreq_option_create ()` se používá pro vytvoření RREQ zprávy, pro vytvoření RREP zprávy se používá funkce `aodv_pkt_support_rrep_option_create ()` a pro RERR zprávu se používá funkce `aodv_pkt_support_rerr_option_create ()`. Dále jsou zde definovány funkce pro práci s pamětí pro jednotlivé AODV zprávy.
- aodv_request_table.ex.c - zde jsou definovány funkce pro tabulky žádostí pro nalezení trasy.
- aodv_route_table.ex.c - zde jsou vytvořeny funkce pro směrovací tabulky.
- aodv_support.ex.c - v tomto externím bloku kódu jsou definovány funkce pro globální statistiky, odesílání a přijímání aktualizací směrovacích tabulek a tisk směrovacích tabulek.

Kompletní přehled všech souborů, které využívá procesní model aodv_rte, je zobrazen v diagramu: AODV protokol v prostředí OM (viz. Příloha A).

6 TVORBA MODELU MANET SÍTĚ V PROSTŘEDÍ OM A IMPLEMENTACE SMĚROVACÍHO PROTOKOLU

V této části práce bude popsán návrh modelu MANET sítě a implementace AODV směrovacího protokolu v prostředí OM (obr. 6.1). MANET síť bude vytvořena z 72 mobilních stanic, které budou vhodně rozmístěny tak, aby pokryly vytyčené území a při tom byl stále zachován radiový dosah mezi dvěma uzly. Mezi některými stanicemi (modrá čárkovaná čára) bude vytvořen datový provoz typu IP_G711_Voice (interaktivní hlas).



Obr. 6.1: Model MANET sítě

6.1 Vložení a nastavení jednotlivých prvků modelu

Nejdříve byl vytvořen nový projekt. Byla vybrána položka **File - New...**, zadán název projektu a scénáře, nastavena velikost scénáře na **Campus** o rozměrech 5x5 kilometrů a vybrán soubor objektů **MANET**.

Jednotlivé prvky byly vkládány pomocí tlačítka **Paleta objektů**. Z MANET palety byl vybrán prvek manet_station (Mobile node 1 až Mobile node 72) a objekt RX Group Config (nastavení vysílacích parametrů jednotlivých stanic). Dále byly z palety objektů vloženy objekty Mobility Config, ve kterém se nastavují parametry náhodné mobility, a Mobility Domain sloužící k nastavení oblasti mobility.

Všechny prvky byly vhodně rozmístěny a pojmenovány. Pomocí objektu Mobility domain byla vytvořena oblast (přibližně 2x2 km), ve které se budou jednotlivé uzly pohybovat.

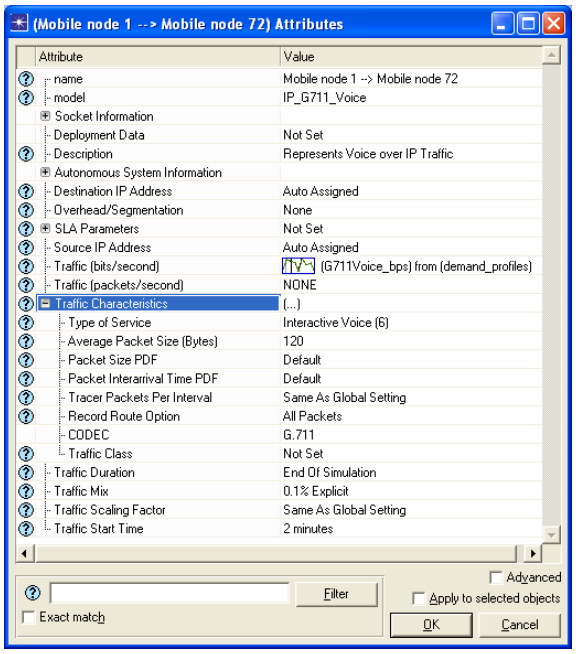
Dále bylo potřeba vytvořit datové provozy mezi některými uzly. Z nabídky jednotlivých provozů v Paletě objektů byl použit model provozu IP_G711_Voice (modrá čárkovaná čára) v položce Demand models.

6.2 Nastavení datového provozu

Aby bylo možné ověřit funkčnost jednotlivých spojení mezi uzly, byl nastaven datový přenos mezi některými uzly. Tyto uzly byly vybrány náhodně. Pro tyto účely byl zvolen typ provozu **IP_G711_Voice**, který je již v prostředí OM předdefinovaný. Parametry provozu byly upraveny a jsou zobrazeny na obr. 6.2. Na modrou čárkovanou čáru datového provozu bylo kliknuto pravým tlačítkem a byla zvolena položka **Edit Attributes**. V záložce **Traffic Characteristics** byly změněny položky **Type of Service** na hodnotu **Interactive Voice (6)** a **Record Route Option** na hodnotu **All Packets** (umožňuje zaznamenat cestu paketů od zdroje k cíli). Dále byly změněny položky **Traffix Mix** na definovanou hodnotu **0,1% Explicit** (původní hodnota „All Background“ neumožňovala záznam hodnot pro zpoždění paketů a kolísání zpoždění) a **Traffic Start Time** na hodnotu **2 minutes** (začátek provozu).

6.3 Nastavení objektu RX Group Config

Objekt Rx Group Config slouží pro omezení možných příjemců, se kterými může každý uzel komunikovat. Parametrů, podle kterých je možné omezit počet příjemců, existuje několik. V našem případě byl zvolen parametr vzdálenosti uzlů. Na objektu



Obr. 6.2: Nastavení datového provozu - IP_G711_Voice

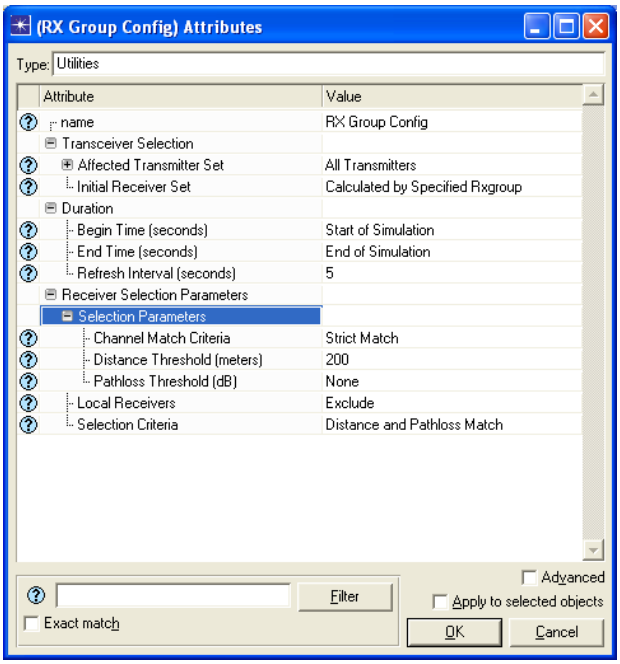
Rx Group Config byla vybrána položka **Edit Attributes** a v záložce **Receiver Selection Parameters - Selection Parameters** byla změněna položka **Distance Treshold (meters)** na hodnotu **200**.

Protože se všechny stanice pohybují, bylo potřeba nastavit hodnotu položky **Refresh Interval** v záložce **Duration**. Tato položka označuje, jak často bude uzel přepočítávat skupinu svých příjemců. Tato položka byla nastavena na definovanou hodnotu **5** sekund, která je pro vytvořený model dostačující. Nastavení objektu Rx Group Config je zobrazeno na obr. 6.3.

6.4 Nastavení objektu Mobility Config a přiřazení náhodného pohybu ke všem uzlům

Pro nastavení náhodného pohybu pro více stanic byl zvolen objekt Mobility Config a na něm byla zvolena položka **Edit Attributes**. V záložce **Random Mobility Profiles** byla nastavena hodnota **Rows** na **1**. Zde byl pouze upraven již nadefinovaný profil **Random Waypoint**. Jeho jednotlivé parametry jsou zobrazeny na obr. 6.4. Mezi tyto parametry patří: název již vytvořené mobilní domény, min./max. vzdálenost na ose x/y (0/500 m), rychlost pohybu (stálá v rozmezí 0-10 m/s), čas pauzy (0 s), začátek/konec pohybu, rychlost aktualizace pozice uzlu a záznam trajektorie.

Všem uzlům byla přiřazena náhodná trajektorie tak, že byl vybrán jeden uzel



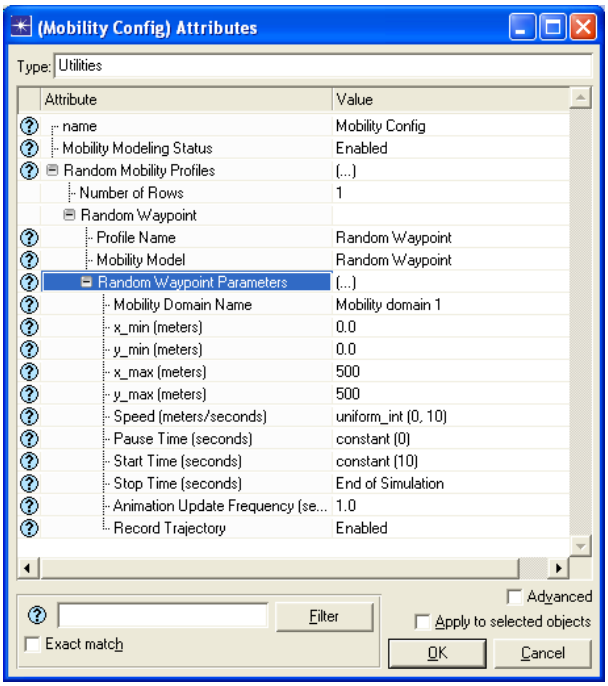
Obr. 6.3: Nastavení objektu RX Group Config

a položka **Select Similar Nodes**. Dále v menu **Topology** - **Random Mobility** - **Set Mobility Profile...** byla zvolena vytvořená mobilita, tedy **Random Waypoint**.

Pokud byl povolen záznam trajektorie (**Record Trajectory** - **Enabled**), je po skončení simulace možné zobrazit jednotlivým uzlům vytvořenou trajektorii pomocí položky menu **Topology** - **Random Mobility** - **Set Trajectory Created From Random Mobility...**.

6.5 Nastavení AODV protokolu a vysílacích parametrů mobilních uzlů

V prvním scénáři byl nastaven AODV směrovací protokol s defaultními parametry. To bylo provedeno vybráním jednoho uzlu a položky **Select Similar Nodes**. Byla zvolena položka **Edit Attributes** - **AD-HOC Routing Parameters** - **AD-HOC Routing Protocol** a zde byla nastavena hodnota na **AODV**. V záložce **AODV Parameters** byla nastavena hodnota **Default**. Ve druhém scénáři byly již parametry AODV protokolu pozměněny. Jednotlivé parametry AODV protokolu byly podrobně prostudovány a byl vyzkoušen jejich vliv na směrování tak, že byly postupně měněny a pomocí simulací byl zjištěn jejich dopad na vlastnosti síťového provozu. Na základě těchto simulací byla snížena hodnota u položky **Net Diameter** na **25**, u položky **Node Traversal Time** byla nastavena hodnota **0,02** a v záložce



Obr. 6.4: Nastavení objektu Mobility Config

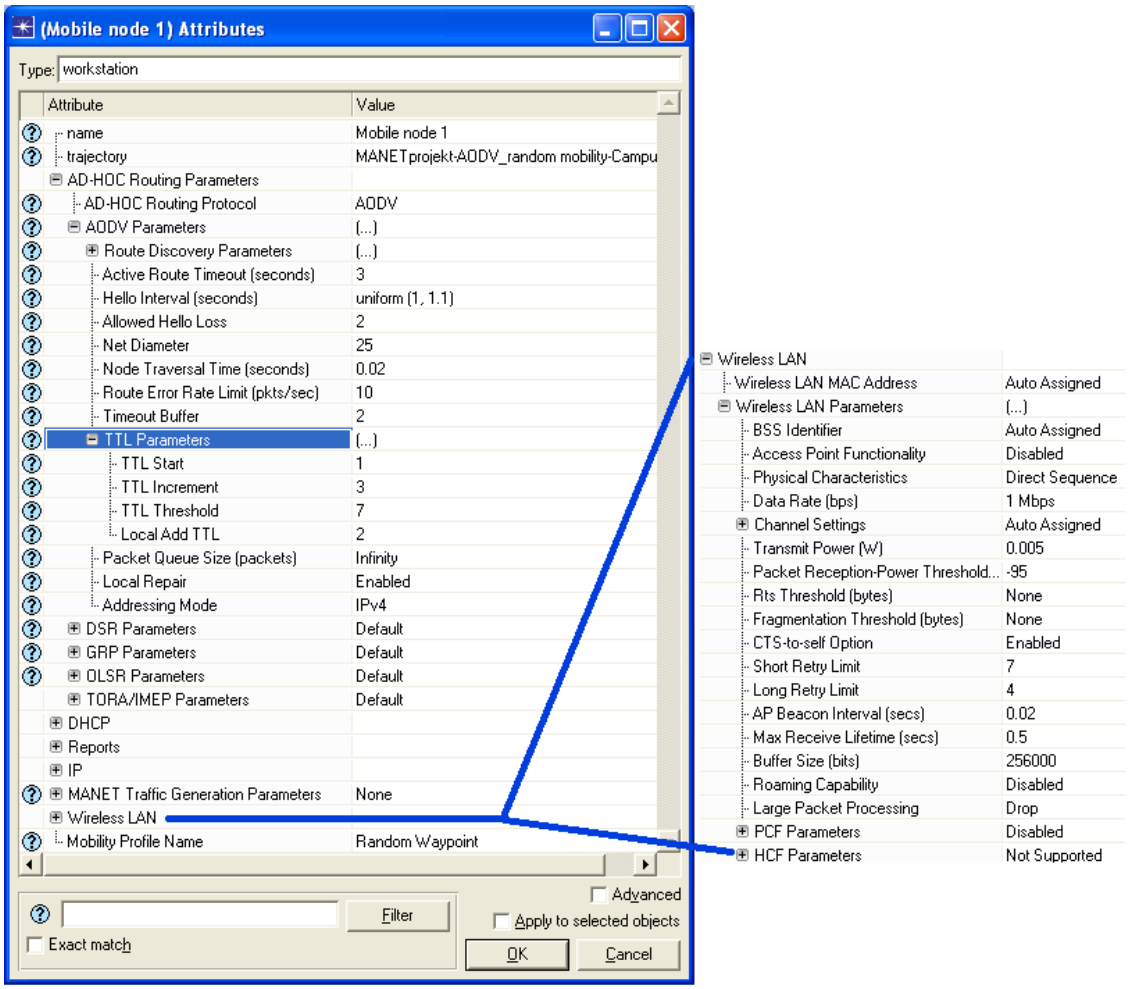
TTL Parameters byla zvýšena hodnota u položky **TTL Increment** na hodnotu **3**. V tab. 6.1 jsou uvedeny jednotlivé parametry AODV protokolu pro oba scénáře. Dále byly nastaveny parametry mobilních uzlů. V položce **Wireless LAN** - **Wireless LAN Parameters** byla snížena hodnota položky **Data Rate** (přenosová rychlost) na hodnotu **1Mbps**. Tímto snížením se dosáhlo reálnější přenosové rychlosti. Nakonec byla zaškrtnuta položka **Apply to selected objects**, aby nastavení změn bylo provedeno na všech uzlech.

Nastavení AODV protokolu je zobrazeno na obr. 6.5 a význam jednotlivých položek je vysvětlen níže:

- Route Request Retries - určuje maximální počet pokusů, kdy se uzel snaží najít cestu posíláním dalších RREQ zpráv.
- Route Request Rate Limit (pkts/sec) - omezuje počet vytvořených RREQ zpráv na jednom uzlu.
- Gratuitous Route Reply Flag - označuje, zda má být „nadbytečná“ RREP zprávy zaslána do uzlu, který je uveden v IP adrese cíle zprávy RREQ, pokud není tento uzel cílový.
- Destination Only Flag - určuje, jestli může být cílový uzel jediný, který odpovídá na RREQ zprávy.
- Acknowledgement Required - pokud je tento příznak povolen musí být každá RREP zpráva potvrzena zprávou RREP-ACK. V současné době není tento atribut podporován.

Tab. 6.1: Hodnoty parametrů pro scénář s defaultními a upravenými parametry

Parametr	**Hodnota**	
	Scénář s defaultními parametry	Scénář s upravenými parametry
Route Request Retries	5	5
Route Request Rate Limit (pkts/sec)	10	10
Gratuitous Route Reply Flag	Disabled	Disabled
Destination Only Flag	Disabled	Disabled
Acknowledgement Required	Disabled	Disabled
Active Route Timeout (seconds)	3	3
Hello Interval (seconds)	uniform(1,1.1)	uniform(1,1.1)
Allowed Hello Loss	2	2
Net Diametral	35	25
Node Traversal Time (seconds)	0.04	0.02
Route Error Rate Limit (pkts/sec)	10	10
Timeout Buffer	2	2
TTL Start	1	1
TTL Increment	2	3
TTL Threshold	7	7
Local And TTL	2	2
Packet Queue Size (packets)	Infinity	Infinity
Local Repair	Enabled	Enabled
Addressing Mode	IPv4	IPv4



Obr. 6.5: Nastavení AODV protokolu a parametrů mobilních uzlů

- Active Route Timeout (seconds) - určuje životnost záznamů směrovací tabulky.
- Hello Interval (seconds) - udává časový interval mezi dvěma HELLO zprávami.
- Allowed Hello Loss - určuje, kolik může být HELLO zpráv ztraceno, než bude spojení prohlášeno za nefunkční.
- Net Diametral - nastavuje maximální možný počet směrování mezi zdrojovým a cílovým uzlem.
- Node Traversal Time (seconds) - udává průměrnou dobu, která je potřebná k přenosu paketů do vzdálenosti jednoho skoku.
- Route Error Rate Limit (pkts/sec) - omezuje rychlost, s jako jsou generovány RERR zprávy. V současné době není tento atribut podporován.
- Timeout Buffer - poskytuje vyrovnávací paměť pro časový limit na obdržení RREP zprávy na odeslanou RREQ zprávu.
- TTL Parameters - nastavuje konkrétní TTL parametry používané během vyhledávání a opravy trasy.
- Packet Queue Size (packets) - určuje počet paketů, které je možno držet ve vyrovnávací paměti při čekání na cestu k místu určení. Pro vysoce mobilní sítě je vhodné použít co největší velikost fronty paketů.
- Local Repair - pokud je tento atribut povolen, bude se uzel snažit o „místní opravu“. To znamená, že se uzel pokusí předat zprávu dál, i když nemá ve své tabulce cestu k cíli.
- Addressing Mode - označuje s jakou verzí Internet Protocolu (IPv4 nebo IPv6) bude AODV protokol pracovat.

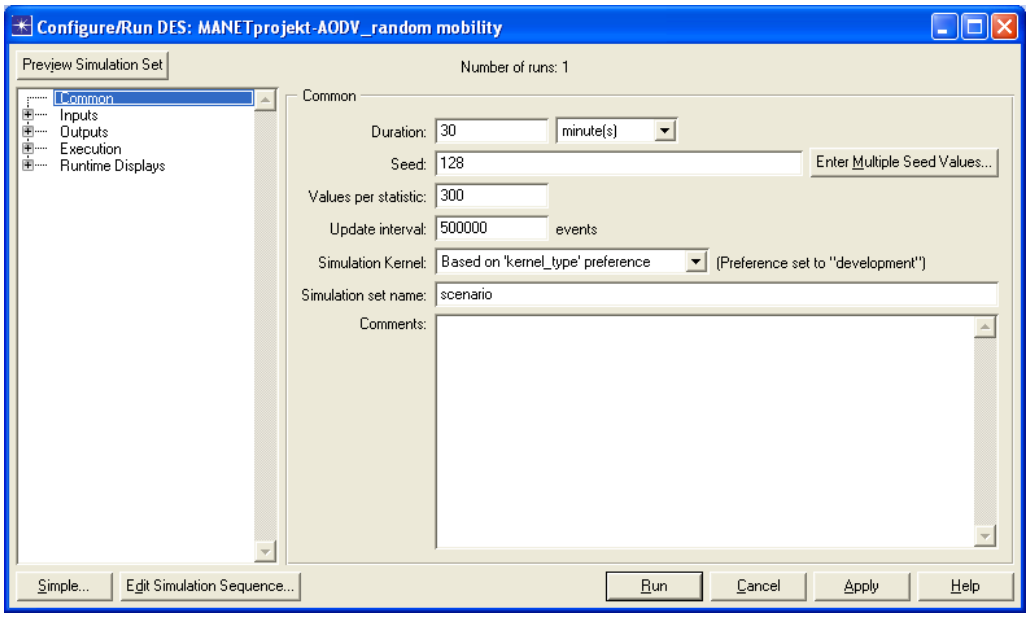
6.6 Vytvoření dalších scénářů

Aby bylo možné porovnat dva scénáře s různými parametry, bylo zapotřebí vytvořit více scénářů modelu MANET sítě. Vytvoření dalšího scénáře bylo provedeno pomocí tlačítka **Scenarios** hlavního menu a položky **Duplicate Scenario...**. Nový scénář byl pojmenován a upraven podle požadavků. Scénáře je možné dále spravovat v položce **Scenarios** - **Manage Scenarios...**.

6.7 Simulace v prostředí OM

Po nastavení všech uzlů, objektů a parametrů podle předešlých kapitol, byly nastaveny statistiky, které budou sledovány. Toto nastavení se provede kliknutím pravým tlačítkem na pracovní plochu prostředí OM a výběrem položky **Choose Individual DES Statistics**. Zde byly vybrány hodnoty AODV směrovacího protokolu a Demand statistiky.

Samotná simulace byla otevřena přes tlačítko „běžce“ pracovní plochy nebo z hlavního menu přes položku **DES - Configure/Run Discrete Event Simulation...**. V otevřeném okně simulace bylo možné nastavit hodnoty položek: **Duration** - doba simulace, **Seed** - počáteční hodnota generátoru náhodného čísla, **Values per statistic** - počet hodnot, který slouží pro vykreslení statistiky, a **Update Interval** - interval, jak často se bude měnit křivka počtu událostí (obr. 6.6). Simulace byla spuštěna tlačítkem **Run**.



Obr. 6.6: Nastavení simulace v prostředí OM

6.7.1 Zobrazení výsledků simulace v prostředí OM

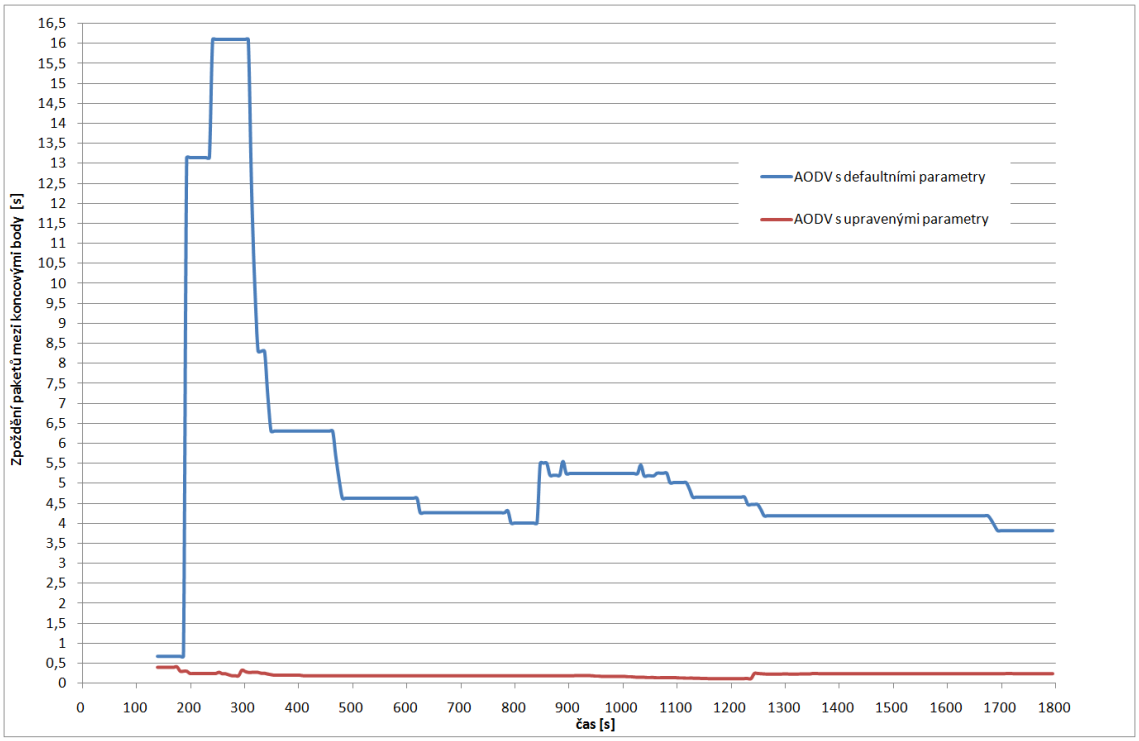
Po skončení simulace byly zobrazeny výsledky simulace kliknutím na pracovní plochu pravým tlačítkem a zvolením položky **View Results**.

Pro zobrazení cest, po kterých probíhalo směrování dat mezi uzly, byla zvolena položka menu **Protocol - IP - Demands - Display Routes for Configured Demands... (DES)**. Zde je možné zobrazit i polohy stanic v daném časovém okamžiku položkou **Show node movement**.

6.8 Výsledky simulace - porovnání parametrů

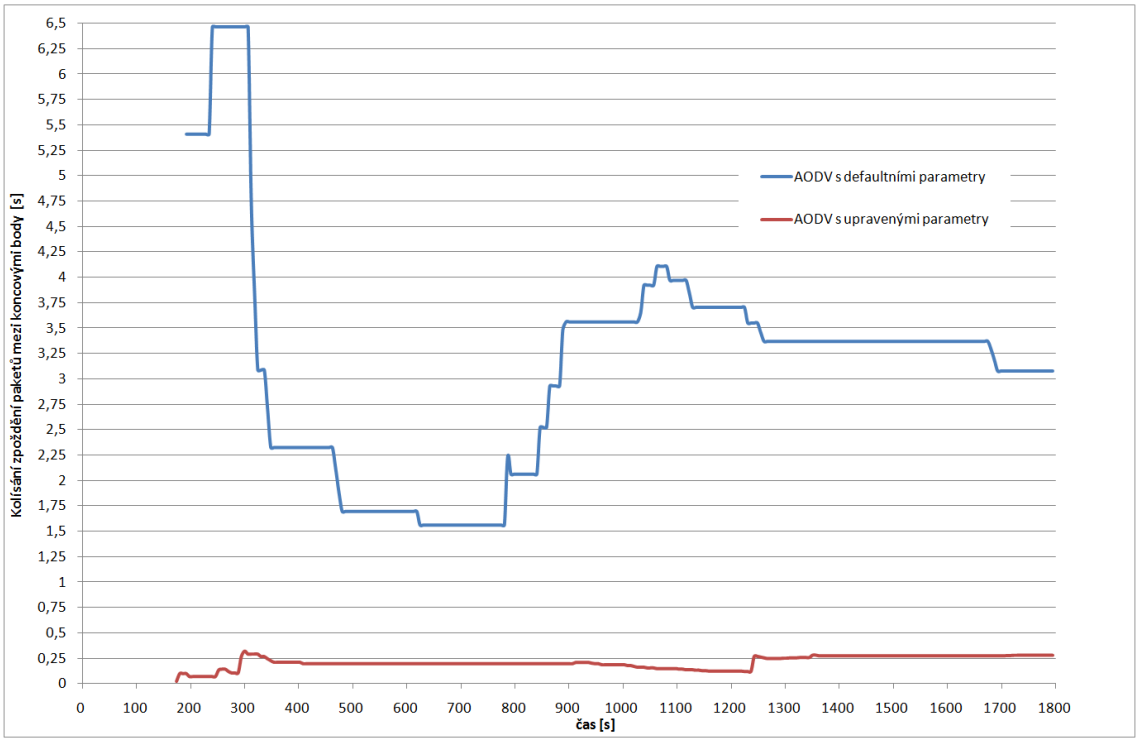
Byl porovnán AODV směrovací protokol z pohledu nastavení jednotlivých parametrů. AODV protokol byl odsimulován ve dvou scénářích. V prvním scénáři byly ponechány defaultní parametry směrovacího protokolu a ve druhém scénáři byly tyto

parametry upraveny, tak aby bylo docíleno co nejlepších vlastností a nedocházelo k většímu zatížení sítě. Po prostudování všech parametrů směrovacích protokolů a jejich vlivů na směrování v MANET sítích byly tyto parametry upraveny tak, jak je popsáno v kapitole 6.5. Na obr. 6.7 je zobrazen graf zpoždění paketů mezi stanicemi Mobile node 22 a Mobile node 65. Z grafu je patrné, jak je možné vhodným nastavením parametrů vylepšit vlastnosti protokolu AODV při zachování téměř stejného zatížení sítě. Při nastavení defaultních parametrů bylo zpoždění paketů průměrně okolo 5 sekund. Při tomto zpoždění by nebylo možné uskutečnit hlasové spojení. Pro interaktivní hlas (IP G711 Voice) je doporučené maximální zpoždění okolo 150 ms. Ve druhém scénáři s upravenými parametry se toto zpoždění povedlo výrazně snížit. Na dalším obr. 6.8 je graf kolísání zpoždění (jitter) paketů mezi stanicemi Mobile node 22 a Mobile node 65. Z tohoto grafu je vidět, jak je vhodným nastavením parametrů možné snížit kolísání zpoždění. Ale i přes tyto vylepšené vlastnosti datového provozu by nebylo možné použít tento směrovací protokol pro aplikace využívající interaktivní hlas. Reaktivní protokoly nejsou obecně vhodné pro přenos aplikací pracujících v reálném čase v takto rozsáhlých sítích.



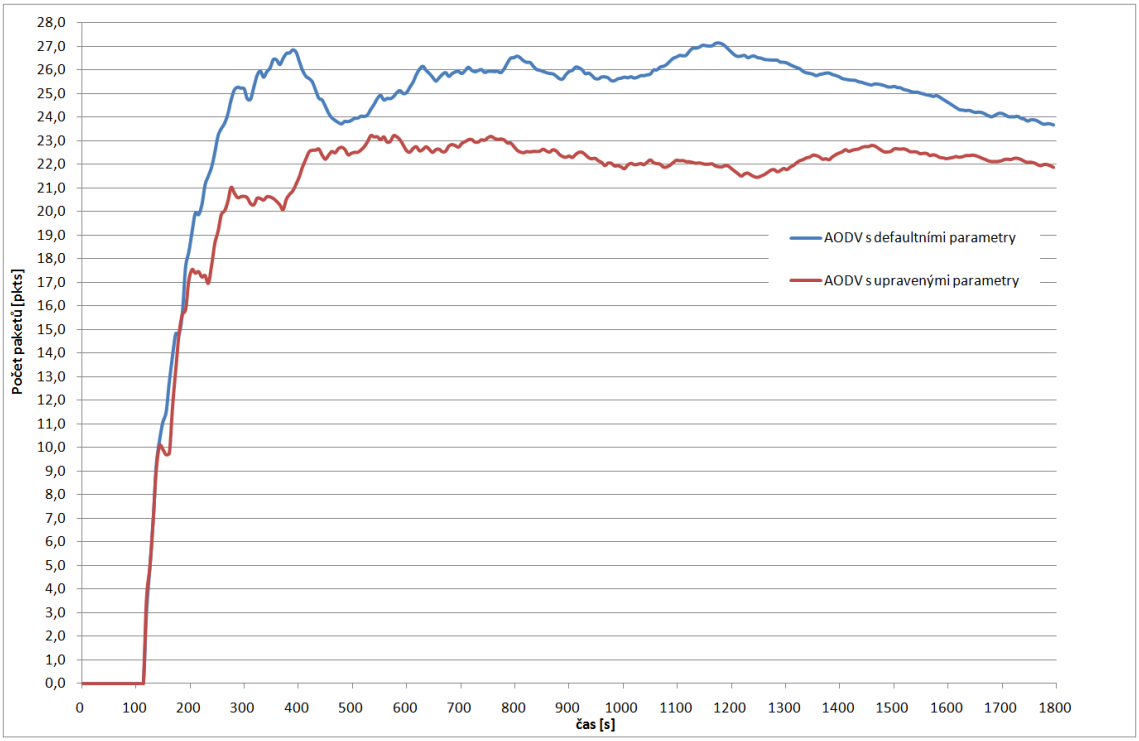
Obr. 6.7: Graf zpoždění paketů mezi stanicemi Mobile node 22 a Mobile node 65

Na posledním obr. 6.9 je graf závislosti odeslaných směrovacích informací v paketech na čase. Na tomto grafu je vidět, že se povedlo vhodným nastavením parametrů snížit nejenom hodnoty zpoždění, ale i snížit zatížení sítě. Dále je z tohoto grafu pa-



Obr. 6.8: Graf kolísání zpoždění paketů mezi stanicemi Mobile node 22 a Mobile node 65

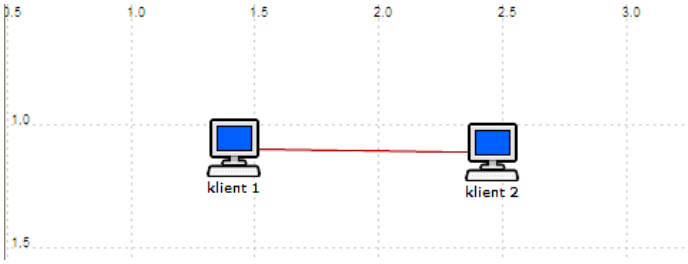
trné, že AODV protokol, který patří mezi reaktivní protokoly, začíná hledat cestu k cíli až při vzniku požadavku. Tedy v našem případě přibližně po uplynutí 2 minut od zahájení simulace, kdy je nastaven start provozu (interaktivní hlas).



Obr. 6.9: Graf závislosti odeslaných směrovacích informací na čase

7 TVORBA MODELU UMOŽŇUJÍCÍHO VYTVOŘENÍ A PŘÍJEM ZPRÁV POMOCÍ ICI

V této části práce bude popsána tvorba procesního modelu umožňujícího vygenerování a příjem zpráv pomocí datové struktury ICI (Interface Control Information), která se v prostředí OM používá k přenosu informací mezi procesy. Výsledný model (obr. 7.1) umožňuje generovat, odesílat a přijímat datové jednotky na síťové vrstvě. Tato vrstva byla zvolena z důvodu implementace AODV protokolu. Tento model byl vytvořen na základě studentské práce kolegy V. Mikulici [5], a to z důvodu seznámení se s úpravami procesních modelů a prací s datovou strukturou ICI.



Obr. 7.1: Model umožňující vygenerování a příjem zpráv pomocí ICI

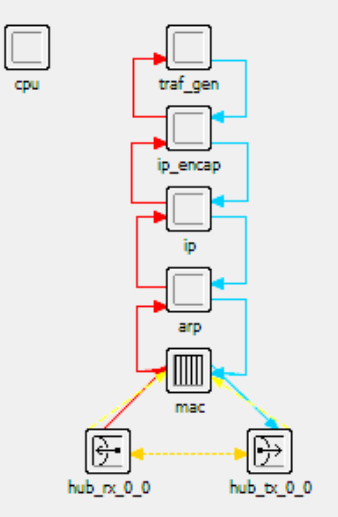
7.1 Vložení a nastavení jednotlivých prvků modelu

Do nově vytvořeného projektu byly vloženy dva prvky ethernet_ip_station_adv (klient 1 a klient 2). Tyto dvě stanice byly propojeny pomocí duplexní linky typu 100BaseT. Každému klientu byla přiřazena IP adresa (192.168.1.1-2). To bylo provedeno volbou položky **Edit Attributes - IP - IP Host Parameters - Interface Information - Address**.

7.2 Uzlový model ethernet_ip_station_adv

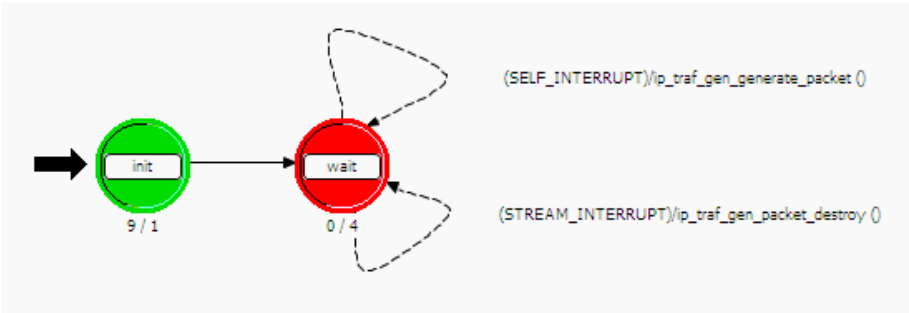
Dvojklikem na klienta bylo vstoupeno do uzlového modelu ethrnet_ip_station_adv (viz. obr. 7.2). Tento model obsahuje tyto moduly: traf_gen, ip_encap, ip, arp a mac. Tyto moduly a jejich procesní modely byly podrobně popsány v práci V. Mikulici [5]. V této práci se zaměříme pouze na moduly traf_gen a ip_encap.

Procesním modelem modulu traf_gen je model ip_traf_gen, který umožňuje přijímat a odesílat IP datagramy. Tento model je tvořen dvěma stavy (obr. 7.3). Výchozí



Obr. 7.2: Uzlový model ethrnet_ip_station_adv

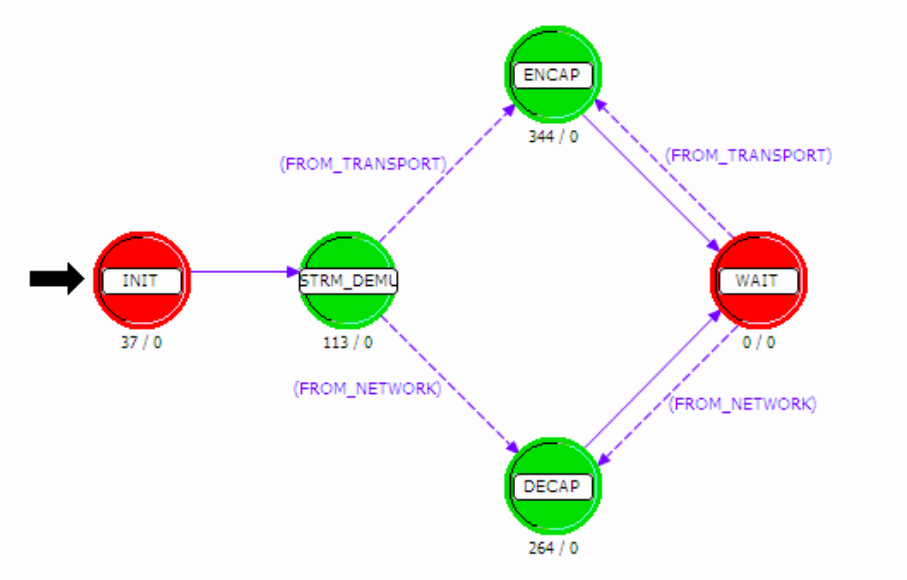
stav *init* inicializuje stavové proměnné, registruje uzel do registru procesů a vyčítá potřebné informace pro nastavení simulace a parametrů provozu. Druhým stavem je stav *wait*, kde proces čeká na příchozí přerušení, tedy vytvoření nebo zničení paketu.



Obr. 7.3: Procesní model ip_traf_gen

Modul ip_encap obsahuje procesní model ip_encap_v4, který zajišťuje zapouzdření přidáním záhlaví IP datagramů přicházejících z vyšších vrstev anebo také odstranění IP záhlaví u paketů přicházejících z nižších vrstev. Tento procesní model je tvořen pěti stavy (obr. 7.4). Ve výchozím stavu *init* probíhá registrace procesního modelu do registru procesů. Dalším stavem je stav *strm_demux*, ve kterém jsou hledány všechny přímo propojené uzly. Proto musí být všechny uzly reprezentující nejbližší vyšší vrstvu zaregistrovány v registru procesů. Z toho stavu může proces přejít do stavu *encap* (datová jednotka z vyšší vrstvy) nebo *decap* (datová jednotka z nižší vrstvy). Ve stavu *encap* je vytvořen IP diagram, do kterého je zapouzdřena datová jednotka spolu s hodnotami jednotlivých polí vyčtených z ICI. Nebo proces přejde do stavu *decap*, kde je z příchozího datagramu vyčtena jeho struktura a je

odeslán vyšší vrstvě. Poslední stav tohoto modelu je stav *wait*, kde proces čeká na příchozí přerušení (příchod datové jednotky).



Obr. 7.4: Procesní model ip_encap_v4

Tento model bude dále upraven pro potřeby použití vlastní datové struktury ICI.

7.3 Úprava a tvorba vlastních modulů a procesních modelů

Ještě před samotnou úpravou jednotlivých modulů/modelů je důležité upozornit, že všechny upravené moduly/modely je potřeba uložit pod jiným jménem a nejlépe do vlastní složky. Toto musí být provedeno z důvodů zachování původních modulů/modelů.

Po vytvoření nové složky, která obsahuje námi vytvořené moduly/modely, je potřeba tuto složku přiřadit do složek modelů. To je provedeno pomocí tlačítka **Edit - Preferences** v záložce M**iscellaneous - Model Directories**. Pokud by byly tyto upravené moduly/modely potřeba přiřadit k jednotlivým modulům/modelům, je potřeba kliknout na modul/model pravým tlačítkem, vybrat položku **Edit Attributes**. Zde zvolit volbu **Advanced** a u položky **model** vybrat příslušný model.

7.4 Úpravy funkčního bloku (FB) procesního modelu ip_traf_gen

Z důvodů implementace vlastní datové struktury ICI, která bude obsahovat nový atribut POLE, bylo potřeba upravit stávající funkční blok (FB) procesního modelu ip_traf_gen. Standardně tento model využívá formát ICI ip_encap_req_v4. Pro naše potřeby byl vytvořen nový formát ICI moje_ici_tm (kapitola 7.5).

```
/** Vytvoření nového ICI **/
  ip_encap_req_ici_ptr = op_ici_create ("moje_ici_tm");
```

Dále bylo potřeba pomocí funkce `op_ima_obj_attr_get()` načíst hodnoty jednotlivých atributů . Tyto hodnoty byly zadány na jednotlivých klientech pomocí položky **Edit Attributes - IP - Traffic Generation Parameters**. Vytvoření těchto nových položek v záložce **Traffic Generation Parameters** je popsáno v kapitole 7.6.

```
/** Načtení hodnoty - atribut TTL **/
  op_ima_obj_attr_get (ith_flow_attr_objid, "TTL",
   &ip_traf_gen_flow_info_array[i].ttl);

/** Načtení hodnoty - atribut POLE **/
  op_ima_obj_attr_get (ith_flow_attr_objid, "POLE",
   &ip_traf_gen_flow_info_array[i].pole);
```

Po vyčtení hodnot atributů bylo potřeba tyto hodnoty přiřadit jednotlivým atributům datové struktury ICI.

```
/** Nastavení hodnoty ICI - atributu TTL **/
  op_ici_attr_set (ip_encap_req_ici_ptr, "TTL",
   ip_traf_gen_flow_info_array[row_num].ttl);

/** Nastavení hodnoty ICI - atributu POLE **/
  op_ici_attr_set (ip_encap_req_ici_ptr, "POLE",
   ip_traf_gen_flow_info_array[row_num].pole);
```

Dalším krokem byla instalace samotné funkce ICI. Poté proces přešel k odeslání datové jednotky a nakonec byla ICI odinstalována s parametrem funkce OPC_NIL (nulová hodnota).

```
/** Instalace ICI **/
  op_ici_install (ip_encap_req_ici_ptr);
```

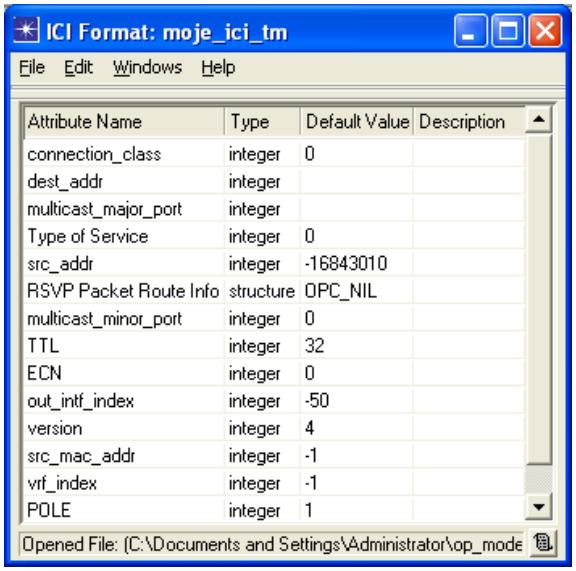
```
/** Odeslání paketu **/
op_pk_send_forced (pkt_ptr, 0);

/** Odinstalování ICI **/
  op_ici_install (OPC_NIL);
```

Nakonec pokud je příchozí přerušení typu stream interupt, bude provedeno vyčtení informací z ICI a také jeho zničení.

7.5 Vytvoření vlastní datové struktury ICI

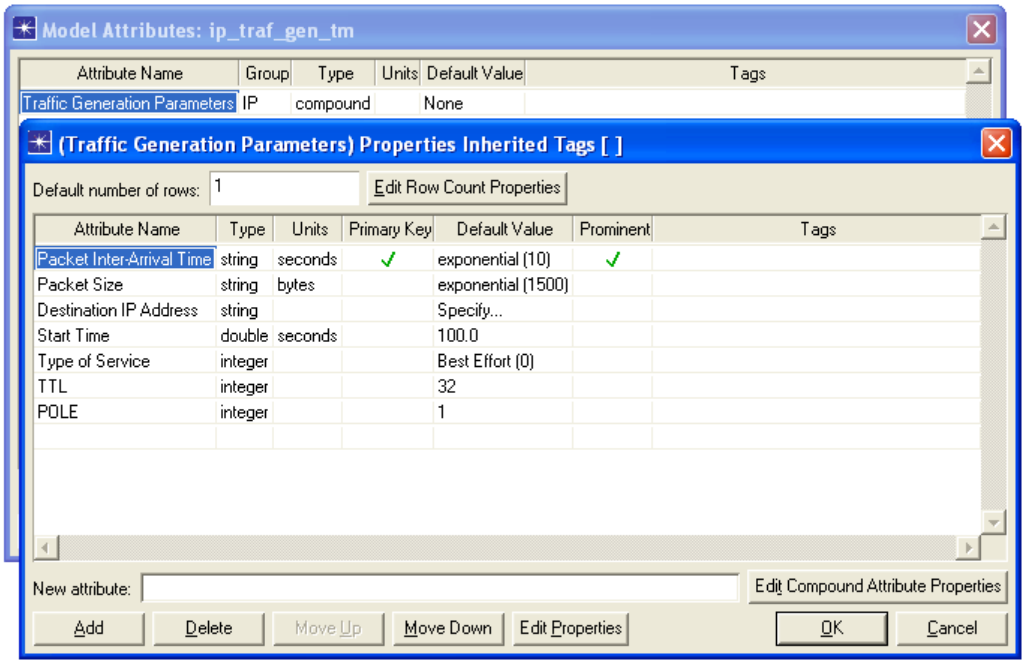
Pro vytvoření vlastní datové struktury ICI byla zvolena položka **File - New** a zde byla vybrána hodnota **ICI Format**. Zde je možné vytvořit nový formát ICI nebo editovat stávající strukturu ICI. V našem případě byl upraven stávající formát ICI ip_encap_req_v4, který byl rozšířen o atribut POLE (obr. 7.5). Jednotlivé atributy ICI formátu ip_encap_req_v4 jsou popsány v práci V. Mikulici [5]. Dále bylo potřeba rozšířit stávající formát ICI inet_encap_req.ic.m o nový atribut POLE. Toto bylo provedeno proto, aby byla zajištěna správná funkce vyšších vrstev s atributem POLE.



Obr. 7.5: Vlastní datová struktura ICI

7.6 Přiřazení nových atributů k atributům procesního modelu ip_traf_gen

Aby bylo možné zadat nově vytvořeným atributům (TTL, POLE) jejich hodnoty, byla zvolena položka **Interfaces - Model Attributes - Edit Properties**. Zde byly přiřazeny nové atributy ke stávajícím atributům procesního modelu ip_traf_gen (obr. 7.6). V našem případě byly přidány atributy TTL (pro možnost zadání životnosti datagramu přímo v nastavení klienta) a POLE (možnost zadat hodnotu pro nový atribut POLE).



Obr. 7.6: Nastavení atributů procesního modelu ip_traf_gen

7.7 Přiřazení nových atributů do datové struktury IP datagramu

Po vytvoření vlastní datové struktury ICI bylo potřeba rozšířit datovou strukturu IP datagramu. Nejprve bylo nutné deklarovat nové pole hlavičky do hlavičkového souboru ip_dgram_sup.h.

```
/** Vytvoření vlastního pole v IP datagramu **/
typedef struct
    {
```

```
    ...

    int pole;
    } IpT_Dgram_Fields;
```

Dále bylo potřeba upravit soubor ip_dgram_sup.ex.c, kde je inicializována struktura z hlavičkového souboru ip_dgram_sup.h. A nakonec byl přidán výpis obsahu pole v debuggeru společně s ostatními atributy IP datagramu.

```
IpT_Dgram_Fields*
ip_dgram_fdstruct_create ()
    {
    IpT_Dgram_Fields*          pk_fdstruct_ptr;
    ...
    /* Přidání POLE */
    pk_fdstruct_ptr->pole = 0;

    FRET (pk_fdstruct_ptr);
    }

/** Výpis obsahu POLE do debuggeru **/
 sprintf(temp_str, "pole int %12d (0)", pk_fd_ptr->pole);
 PKPRINT_STRING_INSERT (alloc_str, temp_str, output_list)
```

7.8 Úprava vstupní pozice stavu encap

Stav *encap* modulu ip_encap tvoří rozhraní mez vyšší vrstvou a síťovou vrstvou. Zde probíhá také zapouzdření datové jednotky přicházející z vyšší vrstvy (tedy z modulu traf_gen). Proto bylo potřeba upravit vstupní pozici stavu *encap* o atribut pole.

```
/** Přidání atributu pole **/
int pole;

/** Vrací hodnotu POLE z ICI a tuto hodnotu uloží **/
if (op_ici_attr_get (ul_iciptr, "POLE",
     &pole) == OPC_COMPCODE_FAILURE)
  {
  ip_encap_error ("Nelze ziskat hodnotu POLE z ICI.");
  }
```

```
/** Získanou hodnotu pole z ICI přiřadí IP datagramu **/
ip_dgram_fd_ptr->pole  = pole;
```

7.9 Simulace

Po nastavení klientů a provedení všech změn bylo možné spustit simulaci přes tlačítko „běžce“ pracovní plochy. Zde byla vybrána položka **Executions - OPNET Debugger** a byla zvolena volba **Use OPNET Simulation Debugger (ODB)**, aby bylo možné při simulaci zachytit IP datagramy rozšířené o atribut pole (obr. 7.7).

Pomocí ODB byly zachyceny IP datagramy, které jsou posílány mezi dvěma klienty a obsahují námi vytvořený atribut pole. Tento atribut byl na straně klienta 1 naplněn hodnotou 1234 (obr. 7.7) a na straně klienta 2 byl naplněn hodnotou 4321. Hodnota TTL byla nastavena na defaultní hodnotu, tedy 32.

```
Index Name                 Type        Value            Size
----- -------------------- ----------- ---------------- ----
    0 fields               structure   0x02B55B58       160
          version            int         4                (4)
          orig_len           int         2320    (16)
          ident              int         1                (16)
          frag_len           int         1480    (13)
          ttl                int         32               (8)
          src_addr           ip_addr     192.168.1.1      (32)
          dest_addr          ip_addr     192.168.1.2      (32)
          protocol           int         500 "ip_traf_gen" (8)
          frag               int         1                (0)
          offset             int         0                (12)
          tos                int         0                (6)
          CE                 int         0                (1)
          ECT                int         0                (1)
          connection_class   int         0                (0)
          src_internal_addr  int         0                (0)
          dest_internal_add  int         1                (0)
          comp_method        comp_info   Not Used         (0)
          original_size      int         160     (0)
    pole int          1234 (0)
          Other fields take up the remaining 23 bits.
    1 options              structure   -                0
    2 data                 packet      pk id (3)        0
    3 MPLS Shim Header     structure   -                0
    4 MPLS Info            structure   -                0
```

Obr. 7.7: Rozšířený IP datagram zachycený v prostředí OM

8 ROZŠÍŘENÍ SMĚROVACÍHO PROTOKOLU AODV O NOVÝ TYP ZPRÁVY

V následujících kapitolách bude popsáno rozšíření směrovacího protokolu AODV o nový typ zprávy. Dále zde bude uveden postup jak vyčítat aktuální přenosovou rychlost ze statistiky a jak tuto hodnotu vložit do nově vytvořené zprávy.

Výstupem bude model MANET sítě s nastaveným AODV protokolem, který bude rozšířen o novou zprávu. Tato zpráva bude zasílána pouze sousedním uzlům a to v pravidelných intervalech. Uzly identifikují novou zprávu a zpracují její obsah. Každá stanice vytvoří soubor, do kterého uloží čas, kdy byla přijata nová zpráva, název uzlu, od kterého byla přijata nová zpráva, a jeho aktuální přenosovou rychlost.

Po prostudování AODV protokolu a jeho implementaci v prostředí OM, bylo rozhodnuto, že nová zpráva bude vytvořena podobně jako HELLO zpráva. Tato zpráva se chová tak, jak je požadováno (v pravidelných intervalech je zasílána sousedním uzlům).

8.1 Úpravy hlavičkových souborů a externích bloků kódů procesního modelu aodv_rte

V prostředí OM je HELLO zpráva tvořena pomocí datové struktury RREP zprávy. Z toho důvodu bylo potřeba nejdříve rozšířit datovou strukturu RREP zprávy (AodvT_Rrep) o nový atribut. To bylo provedeno v souboru aodv_pkt_support.h, kde byl přidán atribut pole typu double. Typ double nebyl zvolen náhodně, ale z důvodu toho, že funkce pro vyčítání statistik `op_stat_local_read()` vrací hodnotu typu double. Tato funkce bude dále použita pro vyčtení hodnoty ze statistiky.

```
/* Route Reply Option */
typedef struct
  {
  ...
  double           pole;
  } AodvT_Rrep;
```

Dále bylo potřeba ve stejném souboru definovat nový typ zprávy. Nová zprávu byla pojmenována AODVC_NEW. Tato zpráva byla definována s indexem 6 (6. zpráva AODV protokolu).

```
/* Type of packet */
#define AODVC_NEW    6
```

Aby bylo možné volat funkci pro vytvoření RREP datové struktury s novým atributem, bylo potřeba rozšířit funkční prototyp v souboru aodv_ptypes.h.

```
/* aodv_pkt_support function prototypes */
AodvT_Packet_Option*
   aodv_pkt_support_rrep_option_create (Boolean, Boolean,
    int, InetT_Address, int, InetT_Address, double, int,
   double);
```

Po rozšíření funkčního prototypu, bylo potřeba upravit i samotnou funkci v souboru aodv_pkt_support.ex.c.

```
AodvT_Packet_Option*
aodv_pkt_support_rrep_option_create (Boolean repair,
   Boolean ack_required, int hop_count, InetT_Address
   dest_addr, int dest_seq_num, InetT_Address src_addr,
   double lifetime, int type, double pole)
  {
  ...
  /* Set the variables of the option */
  ...
  rrep_option_ptr->pole = pole;
  ...
  FRET (aodv_pkt_option_ptr);
  }
```

Dále bylo potřeba v tomto souboru rozšířit vnitřní funkce pro práci s pamětí: `aodv_pkt_support_option_mem_copy ()` a `aodv_pkt_support_option_mem_free ()`.

```
/*** Internally callable functions  ***/
static AodvT_Packet_Option*
aodv_pkt_support_option_mem_copy (AodvT_Packet_Option*
   option_ptr)
  {
  ...
  /* Based on the type of option, copy it */
  switch (option_ptr->type)
    {
      ...
      case (AODVC_NEW):
```

```
      {
      /* This is a route reply option */
      rrep_option_ptr = (AodvT_Rrep*) option_ptr->
         value_ptr;
      copy_option_ptr =
         aodv_pkt_support_rrep_option_create (
         rrep_option_ptr->repair_flag, rrep_option_ptr->
         ack_required_flag, rrep_option_ptr->hop_count,
         rrep_option_ptr->dest_addr, rrep_option_ptr->
         dest_seq_num, rrep_option_ptr->src_addr,
         rrep_option_ptr->lifetime, AODVC_NEW,
         rrep_option_ptr->pole);
      break;
      }
    }
  FRET (copy_option_ptr);
  }

static void
aodv_pkt_support_option_mem_free (AodvT_Packet_Option*
   option_ptr)
  {
  ...
  /* Based on the type of option free it  */
  switch (option_ptr->type)
    {
    ...
    case (AODVC_ROUTE_REPLY):
    case (AODVC_HELLO):
    case (AODVC_NEW):
      {
      /* This is a route reply option */
      rrep_option_ptr = (AodvT_Rrep*) option_ptr->
         value_ptr;
      aodv_pkt_support_rrep_option_mem_free (
         rrep_option_ptr);

      break;
      }
```

```
    ...
    }
  ...
  FOUT;
  }
```

Nakonec bylo potřeba definovat identifikátor nového časovače, který bude použit pro řízení odesílání nové zprávy.

```
/* Constants to identify the timers */
#define AODVC_NEW_TIMER_EXPIRY    6
```

8.2 Úpravy procesního modelu aodv_rte

Další úpravy byly provedeny v procesním modelu aodv_rte. Tyto změny byly provedené v hlavičkovém bloku (HB) a funkčním bloku (FB).

Nejdříve byl upraven hlavičkový blok. Zde byla přidána definice pro časovač NEW_TIMER_EXPIRY a byly zde přidány funkční prototypy pro dvě nové funkce.

```
#define NEW_TIMER_EXPIRY    ((invoke_mode ==
   OPC_PROINV_DIRECT) && (intrpt_type == OPC_INTRPT_SELF)
    && (intrpt_code == AODVC_NEW_TIMER_EXPIRY))

static void     aodv_rte_rrep_new_pkt_arrival_handle (
   Packet*, Packet*, IpT_Dgram_Fields*,
   IpT_Rte_Ind_Ici_Fields*, AodvT_Packet_Option*);
static void     aodv_rte_rrep_new_message_send (void);
```

8.2.1 Úpravy funkčního bloku

Pro získání názvu uzlu (parent_obj_name) bylo potřeba do funkce `aodv_rte_sv_init ()` přidat následující kód, který umožňuje získat hodnotu vlastního ID, ID rodiče a ID ip uzlu.

```
/* Získání vlastního ID */
my_obj_id = op_id_self();

/* Získání ID rodiče */
parent_obj_id = op_topo_parent(my_obj_id);
```

```
/* Získání IP ID */
ip_obj_id = op_id_from_name (parent_obj_id,
   OPC_OBJTYPE_PROC, "ip");

/* Načtení názvu uzlu */
op_ima_obj_attr_get_str (parent_obj_id, "name", 128,
   parent_obj_name);
```

Do funkce `aodv_rte_attributes_parse_buffers_create ()` byl přidán kód pro vytvoření souboru, do kterého se bude ukládat hodnota přijatá v nové zprávě. Každá stanice vytvoří svůj vlastní soubor a pojmenuje ho podle svého názvu (např.: mobile_node_1.txt).

```
char     nazev_uzlu[128];
FILE     *f;

/* Vytvoření souboru a tabulky*/
strcpy(nazev_uzlu, parent_obj_name);
strcat(nazev_uzlu, ".txt");

if ((f = fopen(nazev_uzlu,"w")) == NULL)
{
  printf("Soubor se nepodarilo otevrit\n");
  }

fprintf(f, "
   |--------------|-------------------------------------
---|-----------------------| \n
|   Cas [s]    |                Uzel                    |
  Prenosova rychlost   | \n
|--------------|-------------------------------------
|-----------------------| \n");

if (fclose(f) == EOF)
{
  printf("Soubor se nepodarilo zavrit\n");
  }
```

Aby bylo možné načíst hodnotu pro interval, po kterém bude odeslána nová zpráva, bylo potřeba přidat následující kód.

```
/* Načtení hodnoty NEW paket time*/
op_ima_obj_attr_get (own_mod_objid, "aodv_rte_tm.
   Parameters", &aodv_gen_parms_id);
aodv_gen_parms_child_id = op_topo_child (
   aodv_gen_parms_id, OPC_OBJTYPE_GENERIC, 0);
op_ima_obj_attr_get (aodv_gen_parms_child_id, "NEW paket
   time", &new_paket_time);
```

V procením modelu aodv_rte byly vytvořeny dvě nové funkce. Funkce pro vytvoření a odeslání AODVC_NEW zprávy `aodv_rte_rrep_new_message_send ()` a funkce `aodv_rte_rrep_new_pkt_arrival_handle ()` pro příjem a zpracování AODVC_NEW zprávy.

Nejdříve byla vytvořena funkce pro odeslání nové zprávy. V této funkci dojde k vyčtení hodnoty ze statistiky a poté je testováno, jestli již uplynul nastavený interval od doby, kdy byla naposledy odeslána nová zpráva. Pokud ano, vytvoří se datová struktura AodvT_Rrep, do které se vloží hodnota aktuální přenosové rychlosti. Dále je testováno, jestli se jedná o adresní rozsah protokolu IPv4 nebo IPv6. Poté dojde k vytvoření paketu, do kterého se vloží vytvořená datová struktura. Tento paket je následně zapouzdřen do IP datagramu a odeslán do CPU. Nakonec je uložen čas, kdy byla naposledy odeslána nová zpráva. Zdrojový kód této funkce je uveden v příloze B.

Před vytvořením funkce pro zpracování přijaté nové zprávy bylo potřeba nejdříve rozšířit funkci pro příjem zpráv `aodv_rte_pkt_arrival_handle ()`. V této funkci bylo potřeba přidat kód pro volání nové funkce, která bude zpracovávat zprávu AODVC_NEW na základě hodnoty `tlv_options_ptr->type`.

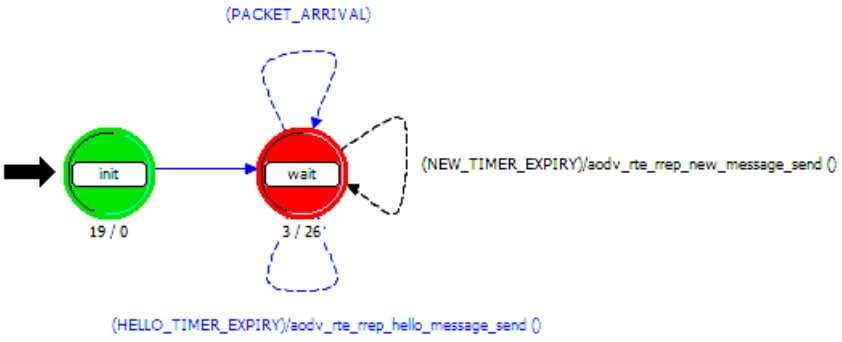
```
switch (tlv_options_ptr->type)
    {
    ...
    case (AODVC_NEW):
      {
      /* Toto je RREP New packet  */
      aodv_rte_rrep_new_pkt_arrival_handle (ip_pkptr,
         aodv_pkptr,
ip_dgram_fd_ptr, intf_ici_fdstruct_ptr, tlv_options_ptr);
      break;
      }
    ...
    }
```

Nakonec byla vytvořena funkce pro zpracování nové zprávy. Tato funkce vyčte hodnotu z položky `rrep_option_ptr->pole` a vytiskne ji do konzole a souboru. Do souboru se uloží čas, kdy byla přijata nová zpráva, název uzlu, od kterého byla přijata nová zpráva, a jeho aktuální přenosová rychlost v bitech za sekundu. Tato funkce je uvedena v příloze C.

8.2.2 Úpravy konečného stavového automatu

V konečném stavovém automatu aodv_rte bylo potřeba vytvořit podmíněný přechod u stavu *wait*, který po uplynutí NEW_TIMER_EXPIRY zavolá funkci pro odeslání nové zprávy `aodv_rte_rrep_new_message_send ()`. Toto rozšíření je zobrazeno na obr. 8.1. Dále bylo potřeba přidat do výstupní pozice stavu *wait* kód, který naplánuje přerušení pro novou zprávu.

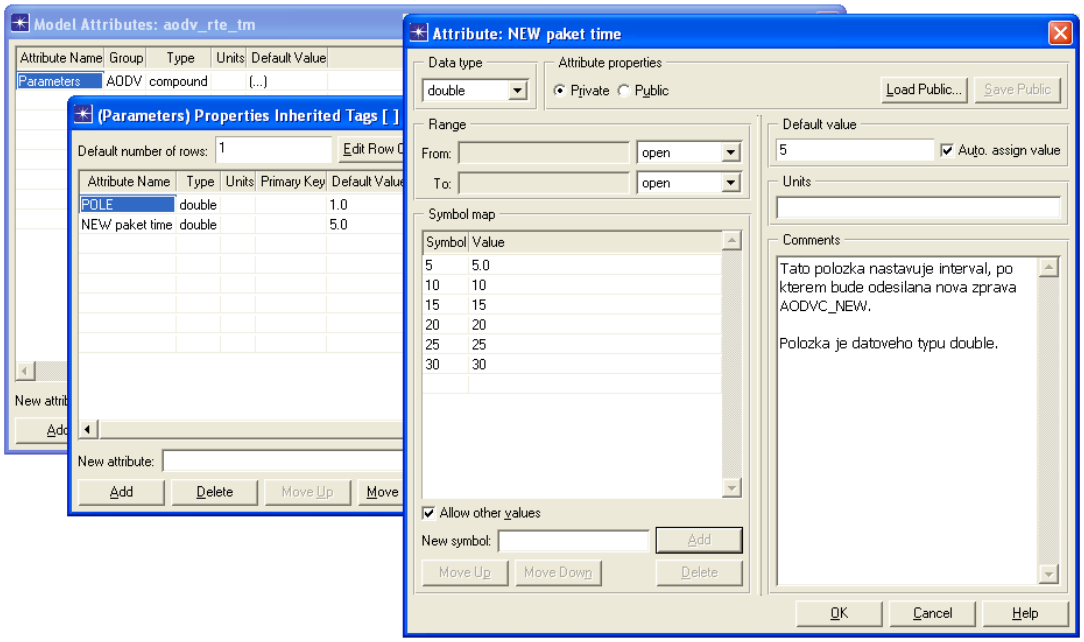
```
op_intrpt_schedule_self (op_sim_time () + new_paket_time,
    AODVC_NEW_TIMER_EXPIRY);
```



Obr. 8.1: Úprava konečného stavového automatu aodv_rte

8.3 Vytvoření nových atributů

Podobně jako v kapitole 7.6 byly vytvořeny nové atributy pomocí položky **Interfaces - Model Attributes - Edit Properties**. Zde byly vytvořeny nové atributy pro interval, po kterém bude odeslána nová zpráva, a pro hodnotu pole, která umožňuje zadat hodnotu atributu pole (viz. obr. 8.2). Intervalu **New paket time** byla nastavena defaultní hodnota **5** a atributu **POLE** hodnota **1**.



Obr. 8.2: Vytvoření nových atributů pro procesní model aodv_rte

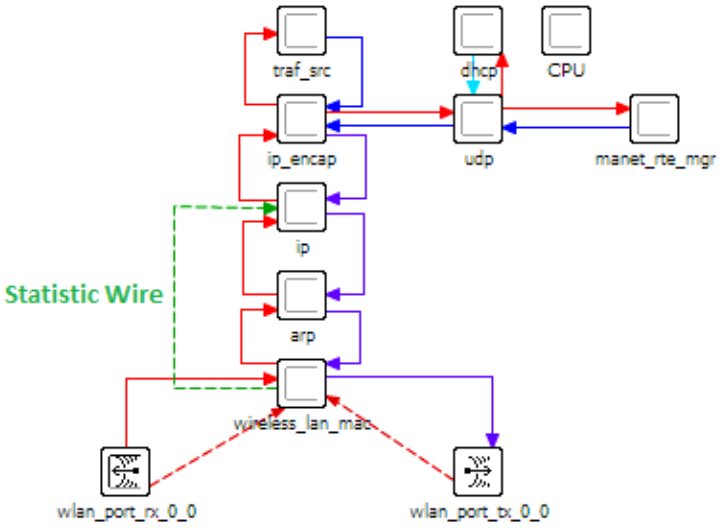
8.4 Úpravy uzlového modelu manet_station_adv

Aby bylo možné vyčítat hodnotu ze statistiky, bylo potřeba vytvořit „Statistic Wire“ mezi modulem, ze kterého chceme vyčítat statistiku, a modulem, ve kterém chceme tuto statistiku zpracovávat, tedy ukládat do nové zprávy.

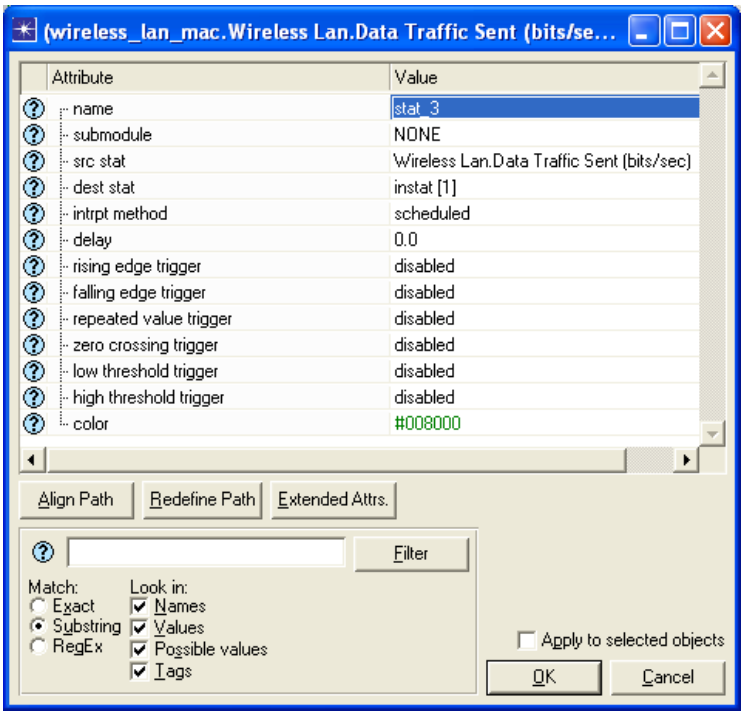
V našem případě bylo potřeba vyčítat hodnotu pro identifikaci aktuální přenosové rychlosti bezdrátového síťového rozhraní MANET stanice. Z tohoto důvodu byla vybrána statistika „Wireless Lan.Data Traffic Sent (bits/sec)“. Tato statistika se nachází na modulu wireless_lan_mac, a proto bylo potřeba propojit moduly ip a wireless_lan_mac pomocí „Statistic Wire“ (obr. 8.3). Vytvořený spoj bylo potřeba upravit pomocí **Edit Attributes**. Zde byla nastavena položka **src stat** na hodnotu **Wireless Lan.Data Traffic Sent (bits/sec)**, položka **dest stat** na hodnotu **instat(1)** a byly zde zrušeny všechny druhy přerušení (viz. obr. 8.4). Položka **dest src** určuje název vstupu statistiky, který je vyčítán pomocí funkce `op_stat_local_read (1)`.

8.5 Úpravy procesního modelu wlan_dispatch

Některé statistiky jsou v OM tvořeny tak, že je hodnota vkládána jen v určitém čase a poté dojde k vynulování statistiky. Z takovýchto statistik je problém vyčítat nenulovou hodnotu v určitém čase.



Obr. 8.3: Úprava uzlového modelu manet_station_adv



Obr. 8.4: „Statistic Wire“ mezi moduly ip a wireless_lan_mac

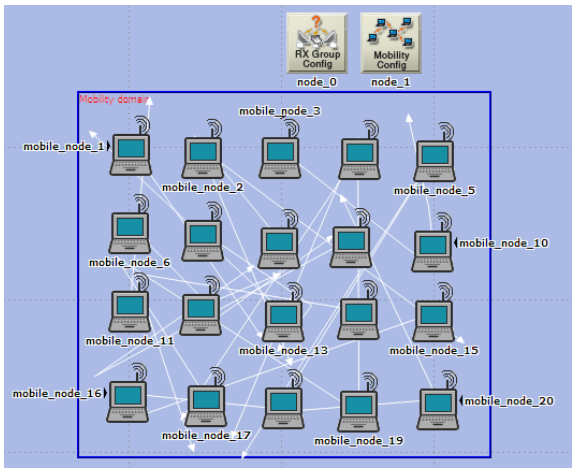
Z tohoto důvodu bylo potřeba upravit podřízený proces modelu wlan_dispatch. Zde bylo potřeba zrušit (zakomentovat) nulování u statistiky „Wireless Lan.Data Traffic Sent (bits/sec)“.

```
/* Zrušení nulování statistiky */
//op_stat_write_t (data_traffic_sent_handle_inbits, 0.0,
   tx_end_time);
```

8.6 Simulace

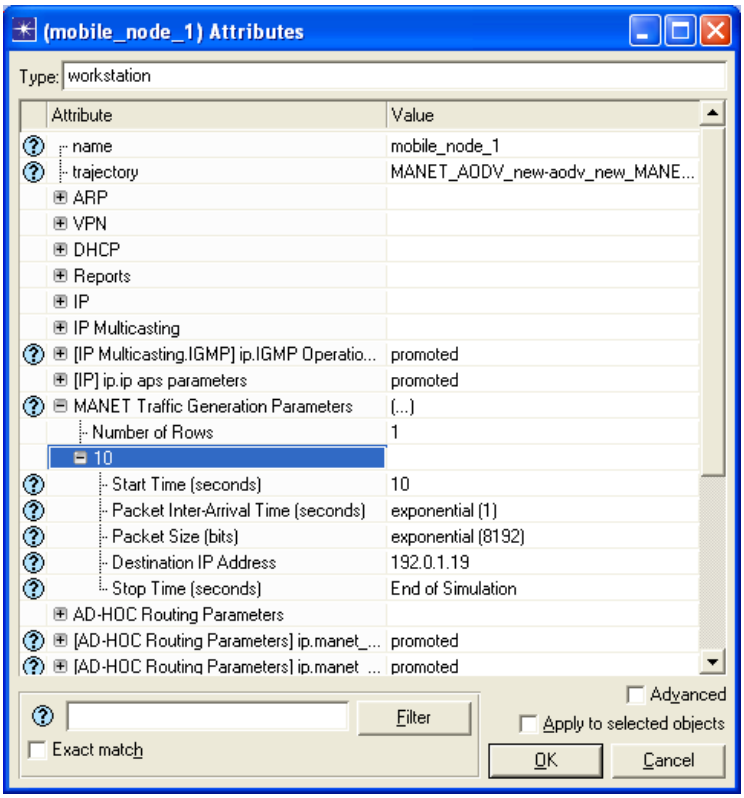
Pro potřeby simulace byl vytvořen model MANET sítě podle kapitoly 6. Výsledný model byl vytvořen pomocí 20 mobilních stanic. Tyto stanice se volně pohybují ve vytyčené mobilní doméně (obr. 8.5). Každé stanici byla nastavena IP adresa pomocí záložky **Protocols - IP - Addressing - Auto-Assign IPv4 Addresses**. Mezi některými stanicemi (mobile_node_1-mobile_node_19, mobile_node_6-mobile_node_10 a mobile_node_14-mobile_node_16) byl nastaven datový provoz pomocí položky **MANET Traffic Generation Parameters**. Datový přenos byl zahájen po 10 sekundách a cílová stanice byla nastavena podle IP adresy (obr. 8.6).

Výhodou OM verze 16 oproti předchozím verzím je možnost mít všechny upravené soubory (hlavičkové soubory, externí bloky kódů a další) u složky projektu aniž by se musela dodatečně nastavovat cesta k těmto souborům.



Obr. 8.5: Model MANET sítě s novou zprávou

Pomocí ODB bylo možné sledovat jednotlivé pakety a výpis do konzole. Během simulace byly vytvořeny soubory jednotlivých stanic. Tyto soubory se nacházejí ve složce projektu. V příloze D je uveden výpis souboru stanice mobile_node_1.



Obr. 8.6: Nastavení datového provozu v položce MANET Traffic Generation Parameters

9 ZÁVĚR

Cílem diplomové práce bylo prostudovat problematiku MANET sítí (viz. kapitola 1). Hlavní pozornost byla věnována zejména procesům směrování (kapitola 2) a směrovacím protokolům AODV a OLSR (kapitola 4 a 3).

Bylo potřeba se seznámit s tvorbou modelu MANET sítě a konfigurací výše zmíněných směrovacích protokolů v prostředí OPNET Modeler. Po nastudování této problematiky byly vytvořeny dva scénáře v prostředí OM (viz. kapitola 6). Každý scénář obsahoval 72 mobilních uzlů, které se náhodně pohybují ve vytvořené doméně, a mezi vybranými uzly probíhá datový provoz - interaktivní hlas. Ve všech scénářích byl nastaven směrovací protokol AODV. První scénář obsahoval defaultní parametry směrovacího protokolu a druhý scénář měl parametry upraveny.

Všechny scénáře byly odsimulovány a výsledky simulací byly popsány v kapitole 6.8. Z těchto výsledků vyplývá, že vhodným nastavením jednotlivých parametrů protokolů se dají ovlivňovat vlastnosti směrovacích protokolů podle předem definovaných požadavků. V našem případě se podařilo snížit zpoždění paketů při zachování stejného zatížení sítě. Bohužel, i přes toto zlepšení není možné tento protokol použít pro aplikace využívající interaktivní hlas.

Dále byl rozšířen procesní model síťové stanice, který umožňuje vygenerování a příjem zpráv pomocí datové struktury ICI. Byla zde vytvořena vlastní datová struktura ICI obsahující nový atribut POLE. Každá stanice je schopna vytvořit, odeslat a přijmout novou datovou strukturu. V této kapitole (kapitola 7) byl vyzkoušen postup při úpravách procesních a uzlových modelů.

V závěrečné části práce (kapitola 8) byl rozšířen AODV směrovací protokol. Byl vytvořen simulační model MANET sítě, kde byl nastaven směrovací protokol AODV a jeho funkční blok byl rozšířen o dvě nové funkce (odeslání a příjem nové zprávy). Nová zpráva může obsahovat hodnotu načtenou z atributů stanice anebo může obsahovat hodnotu aktuální přenosové rychlosti bezdrátového síťového rozhraní MANET stanice. Tato hodnota byla vyčtena ze statistik. Každá stanice v pravidelných intervalech zasílá novou zprávu sousedním uzlům a tím informuje okolní stanice o aktuálním zatížení svého rozhraní. Stanice, která obdrží nový typ zprávy, tuto zprávu zpracuje a uloží vyčtené informace do souboru. Každá stanice si vytvoří svůj vlastní soubor.

LITERATURA

[1] BEDNÁRIK, J. *Modelování komunikace proprietárním protokolem, určeným pro výměnu informací o podporované technologii QoS, v prostředí OPNET Modeler*, Bakalářská práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2007, 77 s.

[2] ILLYAS, M. *The Handbook of Ad Hoc Wireless Networks*. Boca Raton: CRC Press, 2003, ISBN: 0-8493-1332-5, 559 s.

[3] MAHDAL, O. *Směrovací protokoly v sítích s volnou topologií*, Diplomová práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2008, 74 s.

[4] MACHATA, T. *Tvorba pokročilého simulačního modelu technologie DiffServ v prostředí OPNET Modeler*, Bakalářská práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2009, 50 s.

[5] MIKULICA, V. *Generování datových jednotek v prostředí OPNET Modeler*, Diplomová práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2010, 70 s.

[6] MOHAPATRA, P., KRISHNAMURTH, S. *Ad Hoc Networks: Technologies and Protocols*. Boston: Springer Science, 2005, ISBN: 0-387-22689-3, 270 s.

[7] NĚMEC, P. *Samoorganizující se sítě typu MESH*, Diplomová práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2009, 71 s.

[8] NOVOTNÝ, V. *Mobilní směrovací protokoly s podporou IPv6 (MANET)*, Diplomová práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2007, 77 s.

[9] OPNET Modeler: *MANET discrete events simulation* [online]. 2010, [cit. 15. 11. 2010] Dostupné z URL: <https://www.opnet.com/support/des_model_library/manet.html>.

[10] OPNET Modeler: *Documentation Set*, verze 16.0, OPNET Technologies, Inc., 2009.

[11] RFC 2501 - Specifikace MANET sítě: *Mobile Ad hoc Networking* [online]. 1999, [cit. 16. 11. 2010] Dostupné z URL: <http://www.ietf.org/rfc/rfc2501.txt>.

[12] RFC 3561 - Specifikace AODV protokolu: *Ad hoc On-Demand Distance Vector Routing* [online]. 2003, [cit. 16. 11. 2010] Dostupné z URL: <http://www.ietf.org/rfc/rfc3561.txt>.

[13] RFC 3626 - Specifikace OLSR protokolu: *Optimized Link State Routing Protocol* [online]. 2003, [cit. 16. 11. 2010] Dostupné z URL: <http://www.ietf.org/rfc/rfc3626.txt>.

[14] SEHNÁLEK, A. *Sítě s volnou typologií se zaměřením na technologii MANET*, Bakalářská práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2008, 73 s.

[15] ŠIMEK, M. *MSSY - Směrováni dat v sítích WSN*, Prezentace k přednášce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2009, 37 s.

[16] ZEMAN, O. *Implementace simulačního modelu zjednodušené databáze DiffServ - MIB*, Diplomová práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2008, 55 s.

[17] ŽÁČEK, M. *Mobile IPv6 v prostředí OPNET Modeler*, Diplomová práce. Brno: VUT Fakulta elektrotechniky a komunikačních technologií, 2010, 103 s.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ABR Associativity Based Routing protocol

ADV Adaptive Distance Vector

ANSN Advertised Neighbor Sequence Number

AODV Ad Hoc On Demand Distance Vector

ARPANET Advanced Research Projects Agency Network

DARPA Defense Advanced Research Projects Agency – Agentura pro výzkum pokročilých obranných projektů

DB Diagnostic Block

DSDV Destination Sequenced Distance Vector

DSR Dynamic Source Routing protocol

DV Distance Vector

FB Function Block

FSM Finite State Machine

FSR Fisheye State Protocol

GRP Geographic Routing Protocol

HB Header Block

HSLS Hazy Sighted Link State

ICI Interface Control Information

ID IDentification

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

LAN Local Area Network

LS Link State

MANET Mobile Ad-hoc Networks – Mobilní ad-hoc sítě

MID Multiple Interface Declaration

MPR Multipoint relay

ODB OPNET Simulation Debugger

OLSR Optimized Link State Routing

OM OPNET Modeler

OSPFv3 Open Shortest Path First version 3

QMPR QoS MPR – Quality of Service Multipoint relay

QOLSR QoS Optimized Link State Routing

QoS Quality of Service

RERR Route Error

RFC Request for Comments

RREP Route Reply

RREP-ACK Route Reply Acknowledgment

RREQ Route Request

STD State Transition Diagram

SV State Variables

TB Termination Block

TC Topology Control

TORA Temporally Ordered Routing Algorithm

TTL Time To Live

TV Temporary Variables

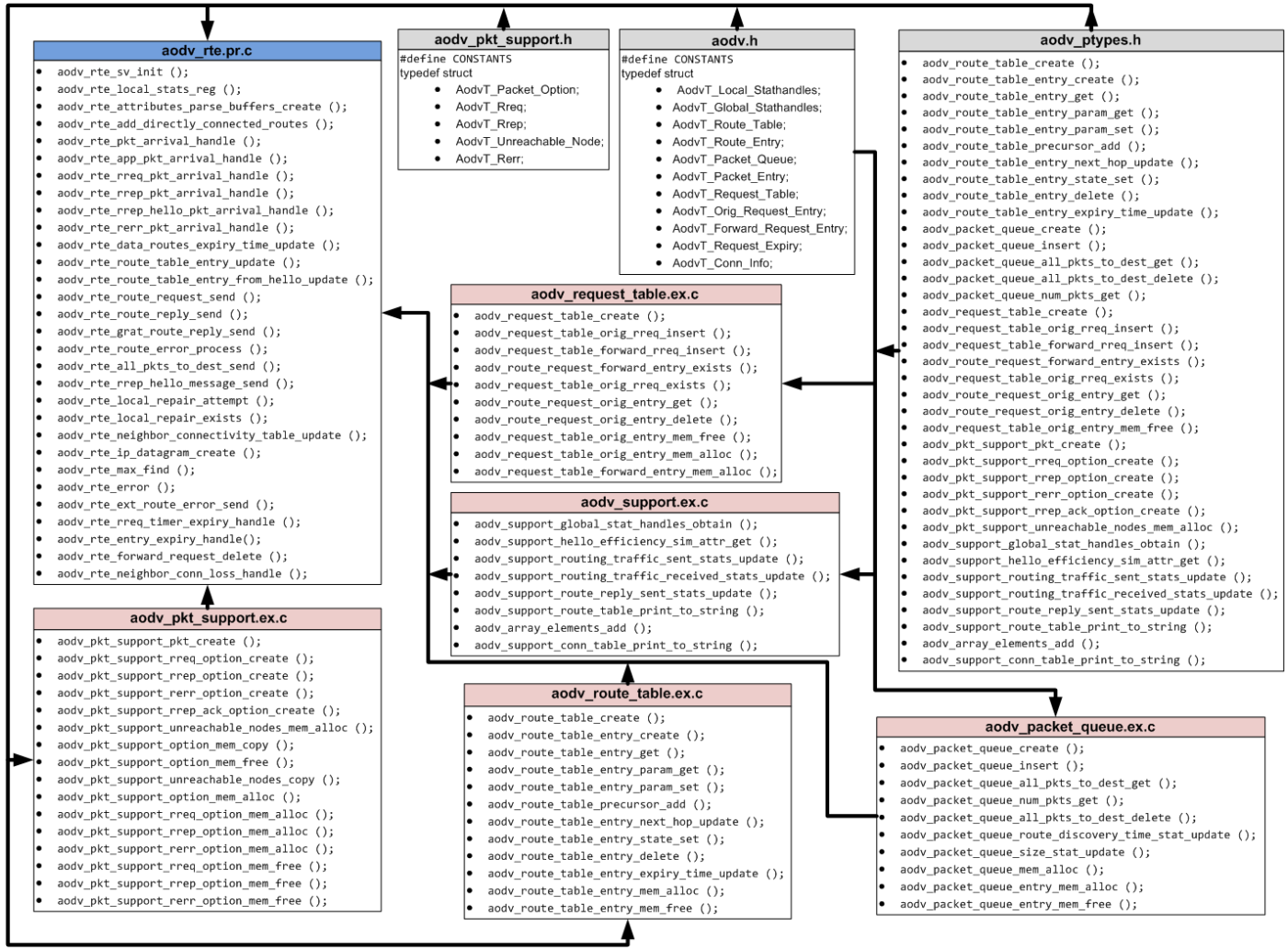
UDP User Datagram Protocol

WRP Wireless Routing Protocol

ZRP Zone Routing Protocol

SEZNAM PŘÍLOH

A DIAGRAM AODV PROTOKOLU V PROSTŘEDÍ OPNET MODELER



B FUNKCE PRO ODESLÁNÍ NOVÉ ZPRÁVY

```
static void
aodv_rte_rrep_new_message_send (void)
  {
  AodvT_Packet_Option*  rrep_option_ptr;
  Packet*       rrep_pkptr;
  Packet*       ip_rrep_pkptr;
  Ici*        ip_iciptr;
  int       mcast_major_port = IPC_MCAST_ALL_MAJOR_PORTS;
  double        akt_rychlost;
  double        akt_stat_rychlost;

  FIN (aodv_rte_rrep_new_message_send (void));

/* Pomocí op_stat_local_read () je zde vyčtena hodnota ze
    statistiky. Hodnota se uloží do proměnné
   akt_stat_rychlost */
akt_stat_rychlost = op_stat_local_read (1);

/*  Podmínka, kdy se testuje, jestli již uběhl interval
   pro odeslání nové zprávy */
  if (((op_sim_time () - last_broadcast_sent_time2) <
     new_paket_time) || (route_table_ptr->
     active_route_count == 0))
    {

    }
  else
    {
    /* Vyčtení akuální přenosové rychlosti z uzlu
       wireless_lan_mac */
    wlan_obj_id = op_id_from_name (parent_obj_id,
       OPC_OBJTYPE_PROC, "wireless_lan_mac");
    op_ima_obj_attr_get_objid (wlan_obj_id, "Wireless LAN
        Parameters", &wlan_par_obj_id);
    wlan_par_obj_id = op_topo_child (wlan_par_obj_id,
       OPC_OBJTYPE_GENERIC, 0);
```

```
    op_ima_obj_attr_get (wlan_par_obj_id, "Data Rate", &
       akt_rychlost);

/* Vytvoření datové struktury AodvT_Rrep */
    rrep_option_ptr = aodv_pkt_support_rrep_option_create
        (OPC_FALSE, OPC_FALSE, 0, INETC_ADDRESS_INVALID,
       sequence_number, INETC_ADDRESS_INVALID,  0,
       AODVC_NEW, akt_stat_rychlost);

    /*  Podmínka, kdy se testuje, jestli se jedná o IPv4
       nebo IPv6 protokol */
    if (aodv_addressing_mode == InetC_Addr_Family_v4)
      {
      /* Vytvoření RREP new IPv4 paketu */
      rrep_pkptr = aodv_pkt_support_pkt_create (
         rrep_option_ptr, AODVC_RREP_IPV4_SIZE);

      /* Zapouzdření AODV paketu do IP datagramu  */
      ip_rrep_pkptr = aodv_rte_ip_datagram_create (
         rrep_pkptr, InetI_Broadcast_v4_Addr, OPC_NIL, 1,
            InetI_Broadcast_v4_Addr, OPC_NIL);
      }
    else
      {
      /* Vytvoření RREP new IPv6 paketu */
      rrep_pkptr = aodv_pkt_support_pkt_create (
         rrep_option_ptr, AODVC_RREP_IPV6_SIZE);

      /* Zapouzdření AODV paketu do IP datagramu*/
      ip_rrep_pkptr = aodv_rte_ip_datagram_create (
         rrep_pkptr, InetI_Ipv6_All_Nodes_LL_Mcast_Addr,
         OPC_NIL, 1, InetI_Ipv6_All_Nodes_LL_Mcast_Addr,
         OPC_NIL);

      /* Vytvoření ICI pro IPv6 paket */
      ip_iciptr = op_ici_create ("ip_rte_req_v4");
      op_ici_attr_set_int32 (ip_iciptr, "
         multicast_major_port", mcast_major_port);
      op_ici_install (ip_iciptr);
```

```
    }

  /* Odeslání paketu do CPU */
  manet_rte_to_cpu_pkt_send_schedule (module_data_ptr,
     parent_prohandle, parent_pro_id, ip_rrep_pkptr);

  /* Nastavení času, kdy byla naposledy odeslána nová
     zpráva */
  last_broadcast_sent_time2 = op_sim_time ();

  /* Odinstalování ICI */
  op_ici_install (OPC_NIL);
  }
FOUT;
}
```

C FUNKCE PRO ZPRACOVÁNÍ NOVÉ ZPRÁVY

```
static void
aodv_rte_rrep_new_pkt_arrival_handle (Packet* ip_pkptr,
   Packet* aodv_pkptr, IpT_Dgram_Fields* ip_dgram_fd_ptr,
    IpT_Rte_Ind_Ici_Fields* intf_ici_fdstruct_ptr,
   AodvT_Packet_Option* tlv_options_ptr)
  {
  AodvT_Rrep*   rrep_option_ptr;
  char       node_name [OMSC_HNAME_MAX_LEN];
  double     hodnota_pole;
  double     cas_simulace;
  char     nazev_uzlu[128];
  FILE     *f;

  FIN (aodv_rte_rrep_hello_pkt_arrival_handle (<args>));

/* Zde se uloží čas, kdy byla přijata nová zpráva */
  cas_simulace = op_sim_time();

  /* Tisk (do konzole) přijaté hodnoty v položce POLE v
     NEW paketu */
  rrep_option_ptr = (AodvT_Rrep*) tlv_options_ptr->
     value_ptr;
  hodnota_pole = rrep_option_ptr->pole;
  inet_address_to_hname (ip_dgram_fd_ptr->src_addr,
     node_name);
  printf("%s: Aktualni prenosova rychlost v case %.3f
     uzlu %s prijata v paketu NEW je: %.2f bitu/s \n",
     parent_obj_name, cas_simulace, node_name,
     hodnota_pole);

  /* Tisk (do souboru) přijaté hodnoty v položce POLE v
     NEW paketu */
  strcpy(nazev_uzlu, parent_obj_name);
  strcat(nazev_uzlu, ".txt");
  if ((f = fopen(nazev_uzlu,"a")) == NULL) {
```

```
  printf("Soubor se nepodarilo otevrit\n");
  }

fprintf(f, " |  %0.3f | %s  | %0.2f bitu/s  | \n",
   cas_simulace,
node_name, hodnota_pole);
fprintf(f, "
   |--------------|-------------------------------------
---|-----------------------| \n");

if (fclose(f) == EOF) {
  printf("Soubor se nepodarilo zavrit\n");
  }

/* Zrušení paketu */
op_pk_destroy (aodv_pkptr);
manet_rte_ip_pkt_destroy (ip_pkptr);

FOUT;
}
```

D SOUBOR STANICE MOBILE_NODE_1

Cas [s]	Uzel	Prenosova rychlost
10.168	Office Network.mobile_node_6	832.00 bitu/s
10.247	Office Network.mobile_node_2	0.00 bitu/s
10.297	Office Network.mobile_node_7	4488.00 bitu/s
15.168	Office Network.mobile_node_6	976.00 bitu/s
15.247	Office Network.mobile_node_2	5512.00 bitu/s
20.168	Office Network.mobile_node_6	10168.00 bitu/s
20.247	Office Network.mobile_node_2	2968.00 bitu/s
20.293	Office Network.mobile_node_8	2968.00 bitu/s
25.168	Office Network.mobile_node_6	832.00 bitu/s
25.247	Office Network.mobile_node_2	13480.00 bitu/s
25.249	Office Network.mobile_node_8	13480.00 bitu/s
25.381	Office Network.mobile_node_11	800.00 bitu/s
30.168	Office Network.mobile_node_6	2672.00 bitu/s
30.247	Office Network.mobile_node_2	832.00 bitu/s
30.249	Office Network.mobile_node_8	2672.00 bitu/s
30.374	Office Network.mobile_node_11	800.00 bitu/s
35.168	Office Network.mobile_node_6	9048.00 bitu/s
35.188	Office Network.mobile_node_13	832.00 bitu/s
35.247	Office Network.mobile_node_2	832.00 bitu/s
35.249	Office Network.mobile_node_8	832.00 bitu/s
35.374	Office Network.mobile_node_11	5848.00 bitu/s
40.168	Office Network.mobile_node_6	800.00 bitu/s
40.188	Office Network.mobile_node_13	800.00 bitu/s
40.250	Office Network.mobile_node_2	6920.00 bitu/s
40.255	Office Network.mobile_node_17	800.00 bitu/s
40.257	Office Network.mobile_node_12	832.00 bitu/s
40.289	Office Network.mobile_node_14	2176.00 bitu/s