BAB II

DASAR TEORI

2.1 Jaringan Komputer

Jaringan komputer adalah sebuah kumpulan dari beberapa komputer yang saling terhubung satu dengan yang lainnya. *Wireless LAN* atau jaringan tanpa kabel adalah suatu bentuk jaringan komputer dimana komunikasi yang terjadi antara perangkat komputer tidak menggunakan kabel sebagai media transmisinya. *Wireless LAN*, menggunakan frekuensi radio sebagai sarana transmisinya, memungkinkan *workstation* dan peralatan *portabel* untuk mengakses jaringan. Sebuah *wireless LAN* terhubung kepada *wired* LAN yang telah ada, memperluas jaringan ke peralatan *mobile computing* yang ada. *Wireless LAN* secara khusus dapat diimplementasikan untuk dalam ruang seperti pabrik, pusat kesehatan, atau kampus, selain itu dapat juga ditempatkan diluar ruangan.

Syarat untuk membangun sebuah jaringan adalah :

- **Physical Connection** : Berhubungan dengan koneksi kartu adapter, seperti NIC maupun modem, ini adalah sesuatu yang dibutuhkan komputer untuk terhubung dengan jaringan. Physical connection digunakan untuk mentransfer signal antara PC dengan jaringan LAN atau juga dengan jaringan yang lebih besar seperti internet.
- **Logical Connection** menggunakan standarisasi, umumnya protokol. Protokol adalah aturan – aturan main yang mengatur komunikasi diantara beberapa komputer didalam sebuah jaringan, aturan ini termasuk didalamnya petunjuk yang berlaku bagi cara – cara atau metode mengakses sebuah jaringan, topologi fisik, tipe –tipe kabel dan kecepatan transfer data. Transmission Control Protocol/internet Protocol (TCP/IP) adalah protokol utama yang harus ada dalam setiap jaringan. TCP/IP digunakan di setiap sistem operasi baik windows, unix maupun machintos.
- **Aplication** atau software program. Aplication menggunakan protokol untuk mengirim dan menerima data melalui jaringan baik LAN maupun internet.

Terdapat dua tipe utama dari struktur jaringan *wireless LAN* mengacu pada *peer-to-peer* (ad-hoc) dan *infrastucture*. Pada struktur jaringan *peer-to-peer* setiap *node* dapat berkomunikasi langsung dengan *node* lainnya dengan asumsi masih berada didalam area jangkauan antar *node*. Pada infrastuktur jaringan *wireless LAN*, semua traffic harus melalui *access point* (AP). Yang Dilakukan dalam transmisi antar dua *wireless node* adalah, yang pertama kali *node* 1 mengirimkan data melalui acccess point dan access point meneruskan data tersebut ke *node* 2, sehingga AP dapat juga berfungsi sebagai relay station. Dari penjelasan diatas AP adalah suatu alat yang dijadikan pusat mekanisme kontrol komunikasi, sehingga jika *node* berada diluar jangkauan access point komunikasi tidak dapat dilakukan.

Teknologi *wireless* LAN ini memiliki keuntungan:

- Memiliki mobilitas dan fleksibilitas yang tinggi.
- Dapat menjangkau area yang luas.
- Dapat diaplikasikan untuk indoor maupun outdoor.

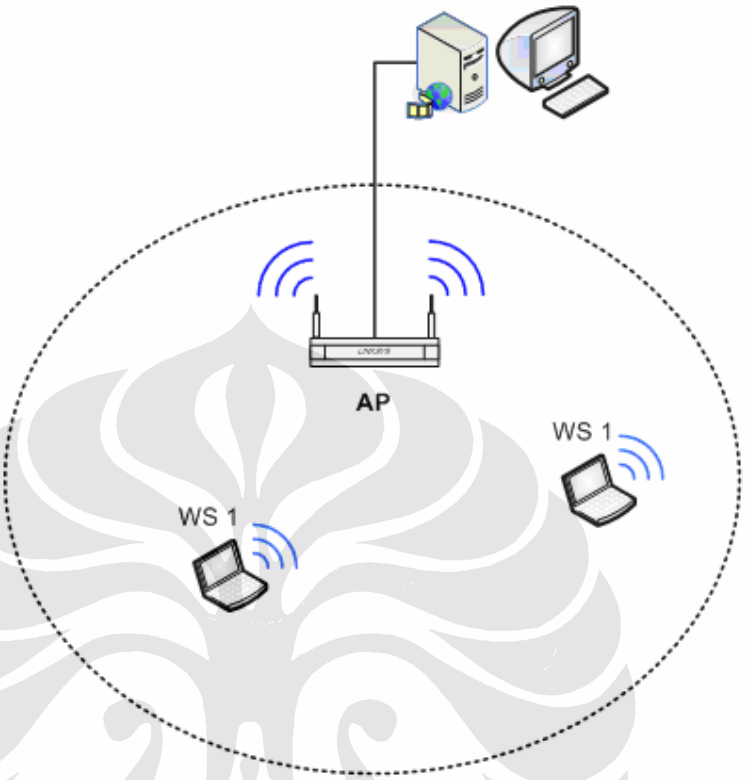
Konfigurasi *wireless* LAN terdiri dari 2 perangkat yaitu:

- *Wireless Station* (WS) : Dekstop, laptop maupun PDA yang dilengkapi dengan *wireless Network Interface Card* (NIC).
- *Access Point* (AP) : Berfungsi sebagai *bridge* antara jaringan LAN konvensional dan WLAN.

Ada 2 jenis mode operasi *wireless LAN* yaitu:

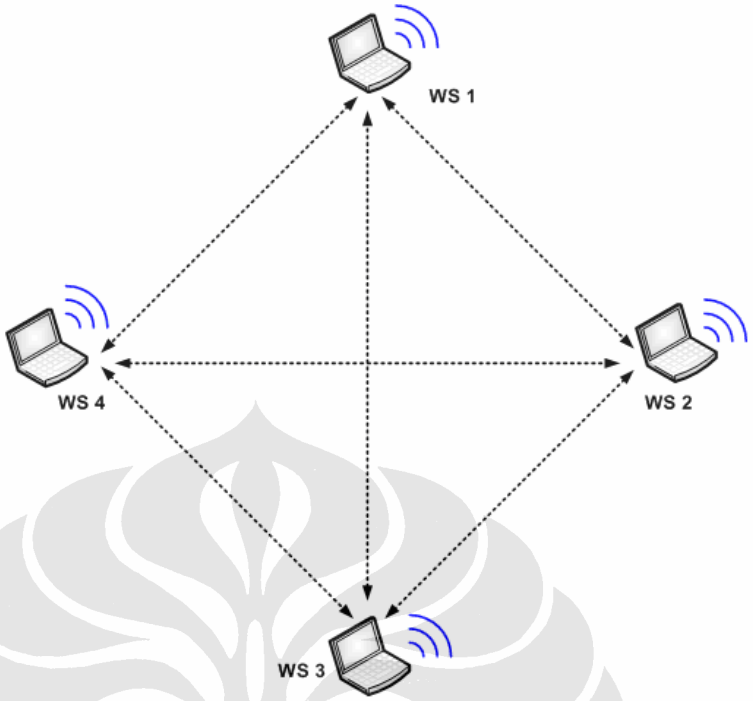
a) Infrastucture Mode seperti ditunjukan pada *Gambar* 2.1 antara lain terdiri dari terdiri dari ; *Basic Service Set* (BSS), Konfigurasi BSS minimal terdri dari sebuah *Access-Point* (AP) yang terhubung ke jaringan kabel atau internet. AP ini dikenal juga sebagai *managed network.* Komunikasi antara dua *station*, misalnya A dan B, harus dari *station* A ke AP kemudian AP mengulang mengirim data ke B. Untuk membangun suatu jaringan dengan server pada konfigurasi ini, server diletakkan pada *Access-Point* dan station-station lainnya sebagai *client.* ; *Extented Service Set* (ESS). ESS terdiri dari beberapa BSS yang saling *overlap* dan masing-masing mempunyai *Acces-Point. Access-Point* satu sama lainnya dihubungkan

dengan *Distributed System* (DS). DS bisa berupa *wired* ataupun *wireless* . DS pada *Gambar* dibawah ini menggunakan kabel.



Gambar 2.1 *Infrastucture Mode*

b) Ad-hoc Mode, terdiri dari beberapa *wireless* station yang berkomunikasi secara langsung (*peer-to-peer*) tanpa menggunakan AP sebagai konfigurasi *independen*. Seperti pada gambar 2.2 dimana semua *user* dapat berkomunikasi secara langsung tanpa adanya perantara. Secara logika berupa *access point* konfigurasi ad-hoc mirip dengan jaringan kabel *peer-to-peer*, dimana komunikasi antar *user* dapat dilakukan secara langsung tanpa adanya *managed network*. Biasanya untuk jaringan *wireless* dalam ruang yang terbatas dan tidak dihubungkan ke jaringan komputer atau internet yang lebih luas.



Gambar 2.2 *Ad-hoc Mode*

2.2 Wireless Mesh Network

Wireless mesh network adalah jaringan komunikasi *wireless* yang terbentuk dari *node* radio dimana minimal terdapat dua atau lebih jalur komunikasi data pada setiap *node* [1]. *Node* pada *wireless mesh network* dapat berupa sebuah *mesh router* ataupun *mesh client*. *Wireless mesh network* memiliki jangkauan yang luas karena setiap *node* tidak hanya bertindak sebagai sebuah *host* tetapi juga dapat berfungsi sebagai sebuah *router* untuk meneruskan paket-paket informasi yang akan dikirim menuju *node* lain yang mungkin tidak dapat menjangkau tempat yang ingin ditujunya karena keterbatasan jarak.

Komponen jaringan *wireless mesh* yang utama adalah suatu perangkat yang selain berfungsi sebagai sumber trafik juga dapat berperan sebagai *router* yang mampu merutekan trafik dari sumber ke tujuan. Perangkat tersebut disebut *wireless mesh node*. Seluruh *wireless mesh node* yang membangun suatu jaringan *wireless mesh* akan bekerja sama untuk membawa informasi dari suatu titik ke titik yang lain. Informasi dibawa dari sumber trafik ke tujuan dengan cara berkerja sama antara *node wireless mesh*.

Apabila terjadi kegagalan pada rute pertama, jaringan dapat melakukan *self healing* sehingga dapat dibentuk rute baru. Salah satu teknologi yang memungkinkan adanya bentuk jaringan *mesh* ini adalah teknologi *wireless* LAN IEEE 802.11. Layanan aplikasi yang dapat disediakan pada jaringan ad-hoc dapat bermacam-macam, mulai dari *Voice Over Internet Protocol* (VOIP), *web browser*, *e-mail*, *media download* dll. Suatu implementasi jaringan *wireless mesh* dengan cara mengelompokkan beberapa *user* ke dalam beberapa *cluster*. Pendekatan ini dilakukan agar trafik ke dalam jaringan *wireless mesh* tidak terlalu besar sehingga tidak terlalu membebani jaringan [2].

Kapasitas dari WMN dipengaruhi oleh banyak faktor seperti arsitektur jaringan, topologi, kepadatan jalur komunikasi, kepadatan *node*, jumlah channel yang digunakan setiap *node*, daya transmisi, dan *mobilitas* dari *node*.

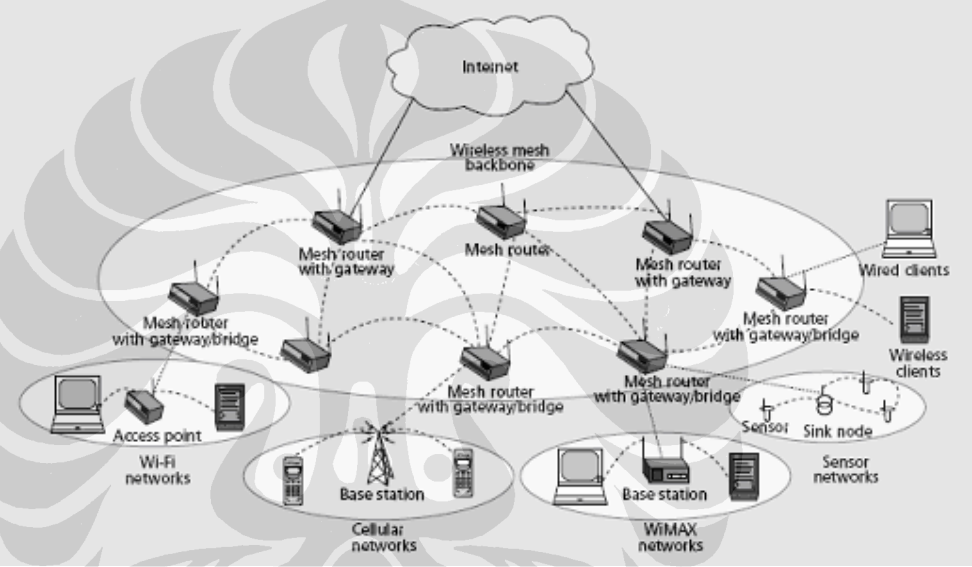
2.2.1 Arsitektur *Wireless* Mesh Network

Wireless mesh networks memiliki dua tipe *node* yaitu: mesh router dan mesh clients. Mesh router selain memiliki kemampuan routing sebagai gateway/repeater seperti fungsi pada *wireless* router konvensional, *wireless* mesh router berisi fungsi routing untuk mendukung jaringan mesh. Lebih lanjut untuk meningkatkan fleksibilitas dari jaringan mesh, biasanya mesh router dilengkapi dengan multiple *wireless interface*. Dibandingkan dengan *wireless* router konvensional, sebuah *wireless mesh router* dapat menerima *covrage* yang sama dengan daya transmisi yang rendah melewati komunikasi *multi-hop*. Medium Access Control (MAC) protokol dalam *mesh router* meningatkan skalabilitas pada area multi-hop. *Wireless mesh* dan konvensional router pada umumnya dibuat berdasarkan *platform hardware* yang hampir sama. *Mesh router* dapat dibuat sebagai tambahan pada komputer, dan dapat dibuat terintegrasi dengan komputer. *Mesh client* juga memiliki fungsi yang penting yaitu sebagai *mesh router*. *Mesh client* biasanya memiliki satu *wireless interface*, sebagai konsekuensi *hardware platform* dan software untuk *mesh client* dapat lebih simpel dari *mesh router*. *Mesh client* memiliki variasi yang lebih banyak dari *mesh router*, antara lain dapat berupa laptop/desktop, pocket PC, PDA, dan IP phone

[4]. Arsitektur dari *wireless mesh network* dapat dikelompokkan dalam tiga kelompok berdasarkan fungsi dari *node* yaitu yang akan dibahas berikut ini.

2.2.1.1 WMN tipe infrastuktur

Bentuk dari arsitektur ditunjukan pada *Gambar* 2.3, dimana garis titik-titik dan garis lurus menunjukan *wireless* dan *wired link*. WMN tipe ini terdiri dari *mesh router* yang terhubung satu dengan yang lainnya membentuk *mesh network*, sedangkan *client* dapat berhubungan dengan *client* lainya melalui *mesh router*.



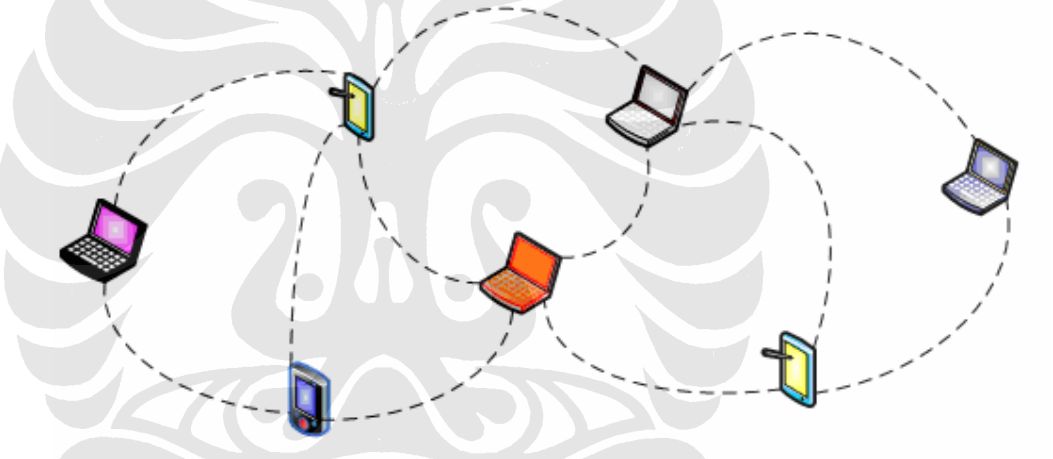
Gambar 2.3 Arsitektur *Wireless Mesh Network* Tipe Infrastruktur

Wireless mesh network jenis infrastuktur dapat dibangun dengan menggunakan beberapa variasi dari teknologi radio dan yang banyak digunakan adalah dengan teknologi IEEE 802.11. *Mesh router* memiliki kemampuan untuk *self-configuring* dan *self-healing* antar *router*. Dengan fungsi *gateway mesh router* dapat terhubung dengan internet. Infrastruktur *mesh* menyediakan *backbone* untuk *client* konvensional dan mengintegrasikan *wireless mesh network* dengan jaringan *wireless* yang telah ada. Komunikasi c*lient* konvensional dengan *ethernet interface* dapat dihubungkan dengan *mesh router* melalui *link ethernet*, selain itu dengan menggunakan teknologi radio yang sama dengan *mesh router client* dapat dihubungkan secara langsung dengan *mesh router*. Jika menggunakan teknologi radio yang berbeda *client* harus menyambung dengan *base station* yang memiliki

koneksi *ethernet* ke *mesh router*. *wireless mesh network* tipe infrastuktur adalah yang paling banyak digunakan saat ini [5].

2.2.1.2 WMN tipe *Client*

Pada *mesh* tipe ini dimana *end-device* (laptop, PDA) berfungsi untuk meneruskan paket yang akan dikirim. *Client* menyediakan komunikasi *peer-to-peer* antar peralatan *client*. Pada arsitektur tipe ini, *client node* merupakan aktual *network* yang melakukan fungsi *routing* dan konfigurasi seperti menyediakan aplikasi kepada pengguna end-device. Dapat dilihat pada gambar 2.4 dimana client dapat berkomunikasi langsung dengan client yang lain tanpa menggunakan mesh router, fungsi *routing* dilakukan oleh client yang akan berhubungan dengan client yang berada diluar jangkauannya.



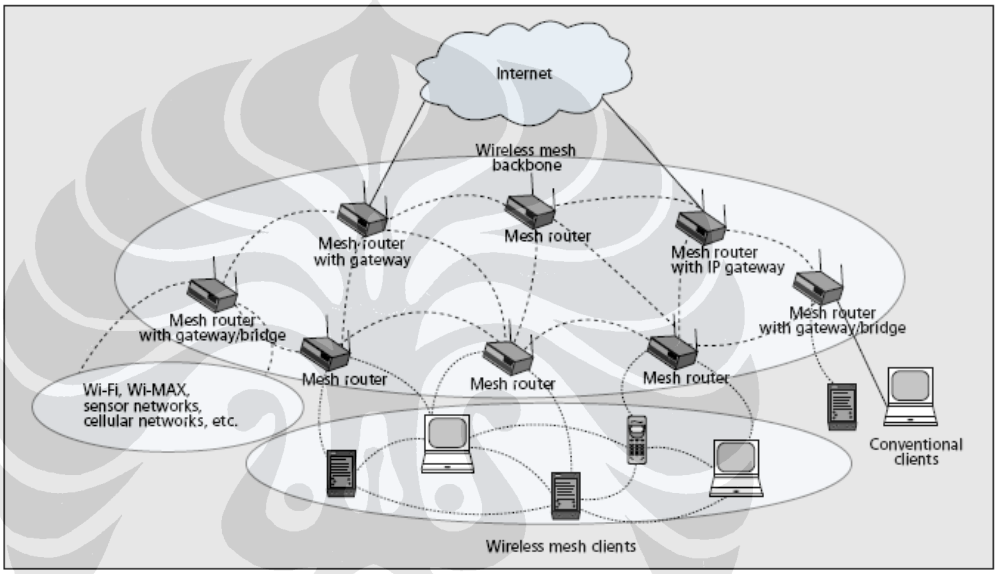
Gambar 2.4 Arsitektur *Wireless Mesh Network* Tipe *Client*

Oleh sebab itu pada tipe ini tidak dibutuhkan *mesh router*. Pada *client wireless mesh* paket ditujukan ke *node* melalui *network* hops melewati multiple *node*s untuk mencapai tujuan akhir, selain itu kebutuhan peralatan pada end-*user* meningkat jika dibandingkan dengan *mesh* tipe infrastuktur, karena *wireless mesh network* tipe *client* harus melakukan fungsi tambahan seperti *routing* dan self configuration

2.2.1.3 Hybrid *Wireless Mesh Network*

Arsitektur ini adalah kombinasi antara WMN tipe infrastuktur dan *WMN* tipe *client*. *Mesh client* dapat mengakses *network* melalui *mesh router* dan dapat

secara langsung berhubungan dengan *client mesh* yang lain seperti pada gambar 2.5 semua *client* dan *router* saling terhubung membentuk jaringan *mesh*. Infrastuktur menyediakan koneksi ke jaringan seperti internet Wi-Fi, WiMAX, *cellular*, dan *sensor network*, kemampuan *routing* dari *client* mampu meningkatkan koneksi dan jangkauan didalam *mesh network*. Arsitektur tipe *hybrid* akan menjadi aplikasi yang paling baik sekaligus menjadi yang paling sulit diterapkan diantara WMN tipe infrasuktur dan WMN tipe *client*.



Gambar 2.5 Arsitektur *Wireless Mesh Network* Tipe *Hybrid.*

2.2.2 Karakterisitik *Wireless* Mesh Network

Ada beberapa karakteristik yang dapat diambil dari penjelasan tiga tipe WMN diatas, seperti berikut ini.

- *Multihop wireless network.* Tujuan dari dibuatnya *wireless mesh network* adalah untuk memperluas *covrage* area tanpa mengurangi kapasitas *channel*. Selain itu untuk menjangkau *user* yang berada dalam *posisi non-line-of-sight* (NLOS).
- Mobilitas tergantung tipe dari *mesh node. Mesh router* pada umumnya memiliki kemampuan mobilitas yang sangat minim, sedangkan *mesh client* dapat bergerak (*mobile*).

- Mendukung untuk jaringan adhoc, *self healing* dan *self organization*. WMN meningkatkan performa dari jaringan, karena arsitektur jaringan yang flexibel, mudah untuk dibentuk dan dikonfigurasi, toleransi kesalahan, komunikasi *multipoint-to-multipont* tergantung dari fitur jaringan.
- *compatibility* dan *interoperability* dengan *wireless network* yang telah ada. Sebagai contoh WMN dibangun berbasis teknologi IEEE 802.11 harus kompatibel dengan standar yang telah ditetapkan oleh IEEE 802.11 yang mendukung keduanya (*mesh* dan *client* Wi-Fi), dan juga WMN harus memiliki kemampuan *inter-operability* dengan jaringan *wireless* seperti WiMAX dan *celullar networks*.
- *Self-Healing*, Jika terjadi kegagalan dalam mengirimkan paket, *node* dapat mencari rute alternatif unuk meneruskan paket yang akan dikirimkan. Kegagalan dalam mengirim paket dapat dipengaruhi oleh interferensi.
- *Self-configuration*, Karena *node* harus mengetahui tetangga terdekatnya sehingga tidak perlu untuk konfigurasi ulang untuk membentuk sebuah jaringan.
- *Reliability*, Pada WMN setiap *node* berfungsi sebagai relay untuk meneruskan paket ke tujuan. Karena *node* dapat masuk dan keluar dari *mesh*, setiap *node* memiliki kemampuan untuk berubah secara dinamis membentuk suatu jaringan berdasarkan *user* yang aktif.

2.3 Lapisan Protokol di Jaringan Komputer.

Secara umum lapisan protokol dalam jaringan komputer dapat dibagi atas tujuh lapisan. Dari *layer* terbawah hingga tertinggi dikenal *physical layer*, *MAC layer*, *network layer*, *transport layer*, dan *application layer*. Masing-masing *layer* mempunyai fungsi masing-masing dan tidak tergantung antara satu dengan lainnya.

2.3.1 *Physical Layer*

Saat ini beberapa teknik dari physical *layer* telah banyak dikembangkan untuk meningkatkan kapasitas dari WMN, antara lain *multiple radio interface*, *multiple-input multiple-output* (MIMO) sistem, *beamforming antenna*, *reconfigurable radios, dan frequency cognitive radio*. Pada *physical layer* yang

utama adalah teknologi radio yang digunakan yang meliputi *data rate physical layer* dan kemampuan beroperasi saat ada *interferensi*. Teknologi yang banyak digunakan saat ini adalah *Code Division Multiple Access* (CDMA), U*ltra Wide Band* (UWB), dan *Othogonal Frequency Division Multiplexing (*OFDM*).* Untuk WMN teknologi yang banyak digunakan saat ini adalah OFDM karena pada teknik ini dapat meningkatkan data rate dari IEEE 802.11 dari 11 Mbps menjadi 54 Mbps, tetapi dengan kepadatan jaringan dan banyak interferensi *data rate* tersebut tidak dapat dicapai. UWB dapat digunakan untuk transmisi data yang besar tetapi kekurangannya adalah hanya dapat digunakan pada jarak yang pendek [6].

Teknik *Orthogonal Frequency Division Multiplexing* (OFDM) merupakan kasus khusus dari FDM (*Frequency Division Multiplexing*). Pada FDM, suatu *bandwidth* dibagi menjadi beberapa kanal yang tersendiri. Agar tidak saling menginterferensi satu sama lain maka diberi jarak antar kanal (*guardband*) yang boros *bandwidth*. Sedangkan dalam OFDM, kanal-kanal dalam satu *bandwidth* seakan-akan ditumpangtindihkan menjadi satu. OFDM sangat efisien dalam penggunaan *bandwidth.*

Spektrum frekuensi kanal pada OFDM dapat ditumpang tindihkan dan tidak terjadi saling interferensi antar kanal, sebab *null* dari setiap kanal yang berdekatan jatuh tepat pada titik tengah spektrum yang membawa informasi (spektrum yang memiliki *power* tertinggi). Untuk mengatur supaya setiap *null* dari kanal spektrum tetangga jatuh tepat pada titik tengah spektrum yang membawa informasi, setiap sinyal transmisi pada setiap kanal harus bersifat saling *orthogonal* dan saling *harmonic.*

2.3.2 *Medium Access Control (MAC) Layer*

Medium Access Control (MAC) untuk WMN lebih ditekankan pada komunikasi *multihop*, MAC berbasis IEEE 802.11 yaitu *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) dengan paket control RTS/CTS (*Ready to Send/Clear to Send*). Fungsi utama MAC *layer* pada WMN adalah untuk mengkoordinasikan akses yang digunakan dengan tujuan memaksimalkan kapasitas *network* agar kapasitas yang diterima dan didistribusikan oleh semua *user* sama besarnya.

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Cara kerja protokol CSMA adalah, suatu *station* yang ingin mentransmisi terlebih dulu memeriksa medium tersebut , jika medium sibuk (misalnya pada saat stasiun lain sedang mentransmisikan data) maka stasiun akan menunda transmisinya hingga medium bebas. Protokol semacam ini sangat efektif jika medium tidak bermuatan penuh, karena memungkinkan stasiun-stasiun mentransmisikan dengan waktu tunda yang singkat, akan tetapi selalu ada kemungkinan stasiun-stasiun mentransmisikan secara bersamaan sehingga kemungkinan besar akan terjadi tabrakan antar paket yang dikirimkan (*collision*), yang disebabkan station mengira bahwa medium telah bebas dan memutuskan untuk segera mentransmisi. Situasi *collision* ini harus diidentifikasi, sehingga MAC *layer* dapat mentransmisikan ulang paket oleh dirinya sendiri bukan oleh *layer-layer* yang lebih tinggi, yang akan menyebabkan penundaan. Dengan DCF, 802.11 sebuah *node* yaitu untuk akses dalam mengirimkan *frame* ketika tidak ada *node* lain menggunakan jalur transmisi, jika ada *node* lain menggunakan jalur transmisi (mengirim frame) station akan menunggu sampai saluran tersebut tidak digunakan. Dalam kondisi mengakses medium, MAC *layer* menggunakan *Network Allocation Vector* (NAV), dimana NAV adalah waktu perkiraan yang akan dipakai oleh *node* yang lain dalam menentukan waktu minimum pemakaian jalur. NAV harus bernilai "0" sebelum sebuah *node* menempati jalur untuk mengirimkan *frame*. Dalam prioritas mengirimkan *frame* , sebuah *node* menghitung jumlah waktu yang diperlukan dalam mengirim *frame* berdasrkan dari panjang *frame* dan *data rate*. Ketika *node* menerima sebuah *frame*, ia akan memerikasa panjang dari *field* yang berguna sebagai basis untuk berhubungan dengan *node* lain [8].

PCF digunakan untuk mengimplementasikan pelayanan terikat waktu (*time-bounded*), seperti transmisi suara atau video. PCF ini menggunakan prioritas yang lebih tinggi yang mungkin diperoleh *Access Point* dengan menggunakan *Inter Frame Space* yang lebih kecil. Dengan menggunakan akses prioritas lebih tinggi, *access point* mengeluarkan permintaan *polling* ke stasiun-stasiun untuk transmisi data, lalu mengendalikan akses medium. Agar memungkinkan stasiun-stasiun regular memiliki kemampuan untuk tetap mengakses medium, Access Point harus memberikan cukup waktu bagi Distributed Access di antara PCF [9].

2.3.3 *Network Layer*

Untuk mengirimkan pesan pada suatu inter*network* (suatu jaringan yang mengandung beberapa segmen jaringan), tiap jaringan harus secara unik diidentifikasi oleh alamat jaringan. Ketika jaringan menerima suatu pesan dari lapisan yang lebih atas, lapisan *network* akan menambahkan *header* pada pesan yang termasuk alamat asal dan tujuan jaringan. Kombinasi dari data dan lapisan *network* disebut "paket". Informasi alamat jaringan digunakan untuk mengirimkan pesan ke jaringan yang dituju, setelah pesan tersebut sampai pada jaringan yang dituju, lapisan *data link* dapat menggunakan alamat *node* untuk mengirimkan pesan ke *node* tersebut.

Meneruskan paket ke jaringan yang dituju disebut "*routing*" dan peralatan yang meneruskan paket adalah "*router*s". Suatu jaringan mempunyai dua tipe *node* antara lain :

- *End node*s, menyediakan pelayanan kepada pemakai. *End node*s menggunakan *network layer* untuk menambah informasi alamat jaringan kepada paket, tetapi tidak melakukan *routing*. *End node*s kadang-kadang disebut "*end system*" (istilah OSI) atau "*host*" (istilah TCP/IP)
- *Router* melakukan mekanisme khusus untuk melakukan *routing*. Karena *routing* merupakan tugas yang kompleks, *router* biasanya merupakan peralatan tersendiri yg tidak menyediakan pelayanan kepada pengguna akhir. *Router* kadang-kadang disebut "*intermediate system*" (istilah OSI) atau "*gateway*" (istilah TCP/IP).

2.3.4 *Transport Layer*

Dua protokol utama pada *layer* ini adalah *Transmission Control Protocol* (TCP) *dan User Datagram Protocol* (UDP). TCP menyediakan layanan pengiriman data handal dengan *end-to-end* deteksi dan koreksi kesalahan. UDP menyediakan layanan pengiriman *datagram* tanpa koneksi (*connectionless*) dan *low-overhead*. Kedua protokol ini mengirimkan data diantara *Application Layer* dan *Internet Layer*. Aplikasi dapat memilih layanan mana yang lebih dibutuhkan untuk aplikasi mereka. Salah satu tanggung jawab *transport layer* adalah membagi pesan-pesan menjadi *fragment-fragment* yang cocok dengan pembatasan ukuran yg dibentuk oleh jaringan. Pada sisi penerima, lapisan transport

menggabungkan kembali *fragment* untuk mengembalikan pesan aslinya, sehingga dapat diketahui bahwa *transport layer* memerlukan proses khusus pada satu komputer ke proses yg bersesuaian pada komputer tujuan. Hal ini dikenal sebagai *Service Access Point* (SAP) ID kepada setiap paket (berlaku pada model OSI, istilah TCP/IP untuk SAP ini disebut port.

2.3.5 *Application Layer*

Pada sisi paling atas dari arsitektur protokol TCP/IP adalah *Application Layer*. Lapisan inilah biasa disebut lapisan akhir (*front end*) atau bisa disebut *user* program. Lapisan inilah yg menjadi alasan keberadaan *layer* sebelumnya. *Layer* sebelumnya hanya bertugas mengirimkan pesan yang ditujukan untuk lapisan ini. Di lapisan ini dapat ditemukan program yg menyediakan pelayanan jaringan, *layer* ini termasuk seluruh proses yang menggunakan *transport layer* untuk mengirimkan data. Terdapat beberapa *application protocol* yang digunakan saat ini. Beberapa diantaranya adalah [10].

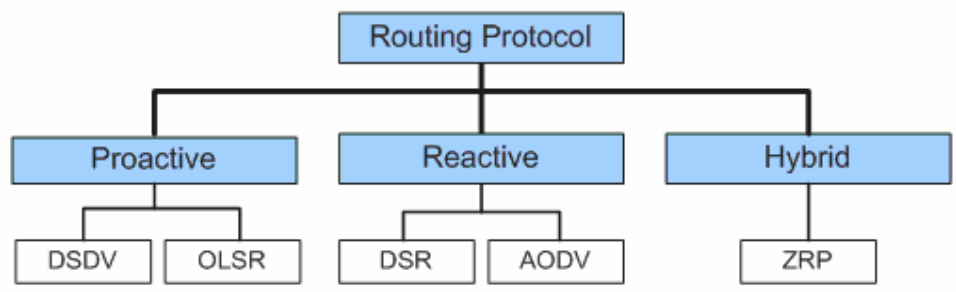
- TELNET, yaitu *Network Terminal Protocol*, yang menyediakan *remote login* dalam jaringan.
- FTP, File Transfer Protocol, digunakan untuk file transfer.
- SMTP, Simple Mail Transfer Protocol, dugunakan untuk mengirimkan electronic mail.
- DNS, Domain Name Service, untuk memetakan IP Address ke dalam nama tertentu.
- NFS, *Network* File System, untuk sharing file terhadap berbagai host dalam jaringan.
- HTTP, Hyper Text Transfer Protokol, protokol untuk web browsing.

2.4 *Routing*

Algoritma *routing* bertanggung jawab menentukan saluran output bagi paket yang akan ditransmisikan. Bila *subnet* menggunakan *datagram* secara internal, saluran harus selalu dibuat baru untuk setiap paket data yang tiba karena rute terbaik mungkin telah berubah pada saat-saat terakhir. Fungsi utama *network layer* adalah merutekan paket dari sumber ke tujuan. Pada sebagian besar *subnet*, paket akan melewati beberapa *hop* untuk sampai ketujuan.

Router memiliki kemampuan melewatkan paket IP dari satu jaringan ke jaringan lain yang mungkin memiliki banyak jalur diantara keduanya. Proses *routing* dilakukan secara hop-by-hop. IP tidak mengetahui jalur keseluruhan menuju tujuan setiap paket. IP *routing* hanya menyediakan IP address dari *router* berikutnya yang menurutnya lebih dekat ke *host* tujuan. *Router* dapat digunakan untuk menghubungkan sejumlah LAN. Jika dua atau lebih LAN terhubung dengan *router*, setiap LAN dianggap sebagai *subnetwork* yang berbeda. *Router* terletak pada *Layer* 3 dalam OSI *refrence* model, *router* hanya perlu mengetahui Net-Id (nomor jaringan) dari data yang diterimanya untuk diteruskan ke jaringan yang dituju. Cara kerjanya setiap paket data yang datang, paket data tersebut dibuka lalu dibaca *header* paket datanya kemudian mencocokkan atau membandingkan ke dalam tabel yang ada pada *routing* jaringan dan diteruskan ke jaringan yang dituju melalui suatu interface. Untuk mengetahui *network* mana yang akan dilewatkan *router* akan menambahkan (Logical AND) *Subnet Mask* dengan paket data tersebut.

Routing menjadi hal yang sangat penting dalam jaringan *mesh*, Protokol *routing* dibagi menjadi tiga bagian seperti gambar 2.6 yaitu: reaktif, proaktif dan hybrid. Protokol *routing* proaktif lebih bersifat *table driven*, dimana setiap *node* menyimpan tabel yang berisi informasi rute semua *node* yang diketahui, informasi rute diupdate secara berkala. Protokol *routing* reaktif adalah on-demand yang berbasis pada sebuah rute yang dibentuk selama permintaan (*request*). *Hybrid routing* protocol adalah kombinasi dari dua protocol *routing* reaktif dan proaktif. Penggunaan protokol *routing* proaktif secara umum memberikan solusi terpendek *end-to-end delay*, karena informasi *routing*, karena informasi *routing* selalu tersedia dan up to date jika dibandingkan dengan protokol *routing* proaktif. Kekurangan dari protokol *routing* proaktif adalah terlalu banyak penggunaan sumber daya (*resource*), seperti *overhead* disaat melakukan *update* informasi *routing* [11].



Gambar 2.6 Klasifikasi Protokol *Routing*

2.4.1 Protokol *Routing* Reaktif

Protokol *Routing* Reaktif berbasis pada sebuah rute yang dibentuk selama *request*, Protokol *Routing* Reaktif, menggunakan pendekatan yang sangat bebeda dengan Protokol *Routing* proaktif. Tabel *routing* dalam protokol ini dibuat apabila terdapat *node* yang akan menggirimkan data. *Node* tersebut melakukan proses *Route discovery* untuk menemukan jalur transmisi paket-paket yang paling optimal. Apabila jalur *routing* telah ditemukan maka tabel *routing* tersebut dipelihara dengan suatu proses route maintenance hingga tidak diperlukan lagi, atau apabila *node* tujuan tidak ditemukan. Keuntungan Protokol *Routing* Reaktif adalah protokol ini memerlukan sumber daya dan traffic yang lebih rendah bila dibanding dengan Protokol *Routing* proaktif. Salah satu contoh dari protokol *routing* reaktif adalah *Ad hoc On-demand Distance Vector* (AODV), *Dynamic Source Routing* (DSR) dan, *Temporally Ordered Routing Algorithm* (TORA).

2.5 Ad hoc On-demand Distance Vector (AODV)

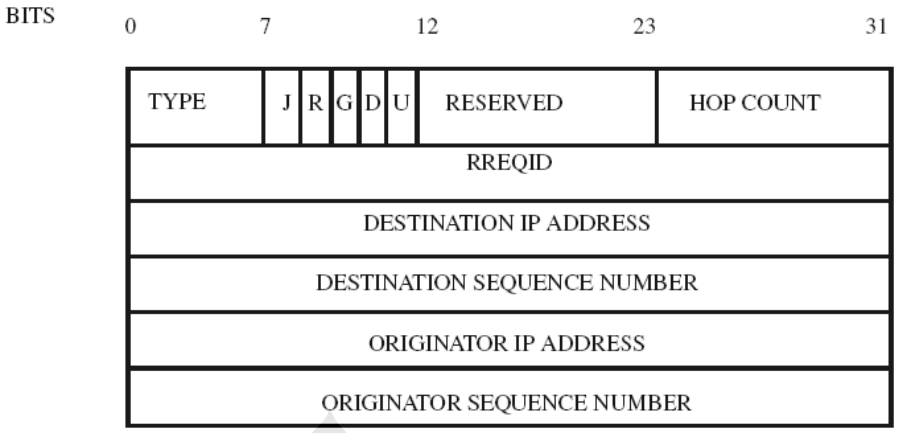
AODV adalah protokol *routing* yang didisain untuk mobile ad-hoc *network*s. AODV memiliki kemampuan routing *unicast* dan *multicast*. Algoritma routing ini berdasarkan permintaan (*on-demand*) artinya rute dibentuk hanya saat terjadinya permintaan dari *node* yang membutuhkannya. AODV dikembangkan oleh C.E.Perkins, E.M. Belding-Royer dan s.Das pada RFC 3561 [12]. AODV sangat simpel, efisien, dan protokol routing yang efektif untuk Mobile-Ad-hoc *network* (MANET) [20]. AODV menggunakan *sequence number* untuk menjamin rute terbaik. AODV membangun rute menggunakan *route request* dan *route reply*. Saat *node* sumber melakukan permintaan rute dimana *node* tersebut tidak

memiliki rute, ia akan melakukan broadcast RREQ ke seluruh jaringan yang terhubung dengannya.

AODV memiliki *route discovery* dan *route maintenance. Route discovery* berupa *route request* (RREQ) dan *route reply* (RREP). *Route maintenance* berupa data dan *Route Error* (RRER) [13]**.** RREQ berjalan dari satu *node* ke *node* yang lain, secara otomatis membentuk jalur untuk kembali dari semua *node* yang di lalui ke sumber *node* yang meminta RREQ. Setiap *node* yang menerima paket RREQ mencatat alamat *node* yang akan menerima RREQ (*destination*), ini biasa disebut *Reverse Path Setup*. *Node* menjaga info selama beberapa saat, untuk RREQ melintasi *network* sampai membuat balasan (*reply*) ke pengirim tergantung dari besarnya *network.*

2.5.1 *Route Request Message*

Pada AODV jalur rute yang dibentuk hanya saat dibutuhkan saja. Protokol AODV mengirimkan sebuah *Route Request* (RREQ) paket menyebar ke seluruh jaringan. Format dari RREQ dapat dilihat pada Gambar 2.7 untuk menguji format pesan dari RREQ menggunakan *sequence number*. *Sequence number* dibuat untuk mengetahui jalur dan informasi rute yang akan dikirim ke *node* tujuan. Jika *node* terdapat dua jalur maka yang dipilih adalah yang memiliki *sequence number* tertinggi atau jalur terpendek. *Sequence number* tertinggi menyatakan sebuah rute terbaru. Untuk melakukan mekanisme pemilihan rute. Ketika terdapat dua kemungkinan, *sequence number* memungkinkan AODV untuk menghindari routing *loop* yaitu paket dikirimkan berulang melalui jalur yang sama, *sequence number* selalu *diupdate* pada saat AODV melakukan *broadcast* RREQ. Setiap *node* yang menerima pesan RREQ memeriksa IP address tujuan, jika *node* tersebut bukan alamat yang dituju maka *node* tersebut segara melakukan *broadcast* ulang pesan RREQ ke *node* terdekatnya sekaligus mengupdate *routing table* yang meliputi *reverse pointer* ke asal pesan. Proses ini terus berjalan hingga menemukan alamat *node* yang dituju atau IP datagram mencapai hop maksimum dalam mengirimkan RREQ [14].



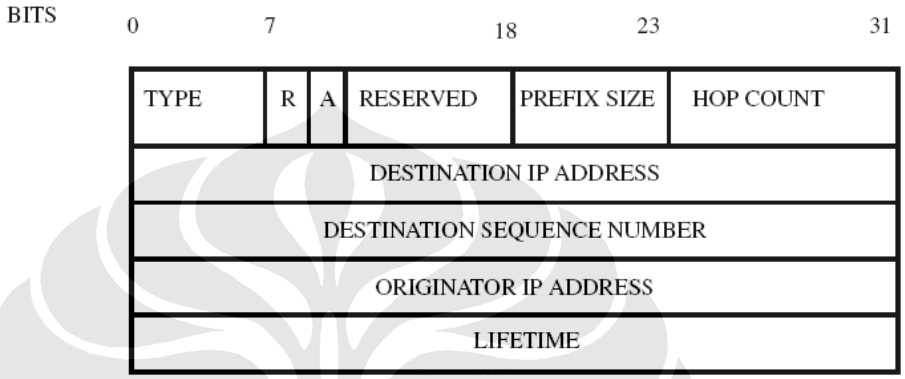
Gambar 2.7 Format Route Request [14]

- J : Join flag.
- R : Repair flag.
- G : Gratuitous RREP flag; mengindikasikan apakah sebuah flag asumsi harus unicast ke *node*.
- D : Destination Only Flag; menyatakan hanya *node* tujuan yang menerima RREQ ini.
- U : Unknown sequence number; menunjukan destination sequence number tidak diketahui.
- Reserved : Sent as 0. abaikan atau terima
- Hop Count : Jumlah dari hop dari IP addres asal ke *node* yang menangani permintaan.
- RREQ ID : Sebuah sequence number unik yang mengidentifikasi RREQ ketika berada di cabang untuk sampai ke addres tujuan.

2.5.2 *Route Reply Message*

Saat pesan RREQ telah sampai ke *node* tujuan maka akan dibentuk rute baru ke *node* asal. AODV mengadopsi mekanisme yang sangat berbeda untuk menjaga informasi *routing*. AODV menggunakan tabel *routing* dengan satu *entry* untuk setiap tujuan. Tanpa menggunakan *routing* sumber AODV menggunakan tabel *routing* untuk menyebarkan RREP kembali kesumber dan secara *sequensial* akan mengarahkan paket data ketujuan. AODV juga menggunakan *sequence number* untuk menjaga setiap tujuan agar didapat informasi *routing* terbaru dan

untuk menghindari *routing loops*. Semua paket yang dikirim membawa *sequence number* ini. Saat membuat RREP sebuah *node* meng-copy IP *address* tujuan dan *Originator Sequence Number* dari RREQ ke *field* yang sesuai pada RREP. RREP bersifat *unicast* ke arah hop yang membuat RREQ, *field Hop count* selalu bertambah saat melewati setiap *node*. Ketika RREP mencapai tujuan, *hop count* merepresentasikan jarak dari tujuan (*destination*) ke sumber (*originator*).



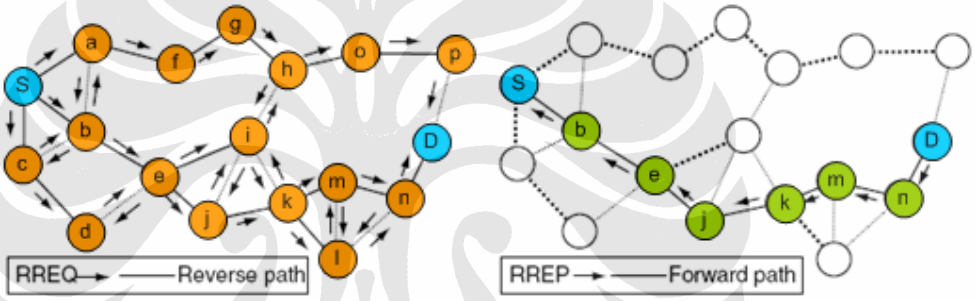
Gambar 2.8 Format *Route Reply* [14]

- R : Repair flag; digunakan untuk multicast.
- A : Acknowledgement required.
- Reserved : Set as 0; abaikan atau terima.
- Prefix Size : Jika bukan zero, 5-bit prefix size menetapkan bahwa hop berikutnya mungkin telah digunakan oleh *node* lain yang menngunakan roting prefix yang sama sebagai tujaun akhir.
- Hop Count : Jumlah hop dari awal ke IP addres tujuan.
- Lifetime : Jumlah waktu dalam milisecond yang valid untuk sebuah *node* menerima RREP.

2.5.3 *Route Discovery*

Route discovery dimulai dengan melakukan *broadcast* pesan RREQ yang berisi alamat tujuan dan *destination sequence number* yang menjamin bebas dari loop, keseluruh jaringan (*Gambar* 2.9). Ketika RREQ masuk ke jaringan setiap *intermediate node* membentuk rute kembali ke sumber (*originator*). Jika sebuah *node* menerima RREQ maka *node* tersebut akan mengirimkan RREQ lagi ke *node* Penentuan jalur dibentuk dengan mengirimkan *route reply* , ketika *route reply*

masuk kestiap *node* ia akan secara otomatis melakukan *setup* jalur. Jika sebuah *node* menerima RREP, maka *node* tersebut akan meneruskan RREP lagi ke *node* tujuan atau *destination sequence number*. Pada proses ini, awalnya *node* memeriksa *destination sequence number* pada tabel *routing*, apakah lebih besar dari satu pada RREQ jika benar maka *node* akan mengirimkan RREP. Saat RREP berjalan kembali ke sumber melalui path yang telah disetup, ia akan mensetup jalur ke depan dan mengupdate time-out**.** Terdapat kemungkinan pada *node* yang mengirimkan RREQ (*originator*) menerima pesan RREP lebih dari satu *node*. Pada kasus ini *originator* akan mengupdate *routing table* dengan informasi *routing* yang terakhir didapat, dan yang akan digunakan adalah yang memiliki *destination sequence number* tertinggi.



Gambar 2.9 Mekanisme RREQ Dan RREP Pada AODV

Karakterisitik AODV

- Minimal space complexity : Hanya *node* tertentu yang menjaga informasi rute, *Node* yang tidak aktif (bukan jalur yang dilalui) tidak menjaga informasi rute. Setelah menerima RREQ dan membentuk jalur kembali dalam routing table dan menyebarkan kembali ke tetangga terdekatnya, jika tidak menerima RREP *node* akan menghapus informasi routing yang telah dicatat.
- Memiliki bandwidth yang besar : Semua intrmediate *node* pada jalur yang aktif mengupdate routing table dan memaksimalkan penggunaan bandwidth, walaupun routing tabel digunakan berulang, dan *intermediate node* menerima RREQ dari sumber yang lain untuk tujuan yang sama.
- Simple : Setiap *node* bertindak sebagai router dan, menjaga routing table.

- Informasi routing yang efektif : Setelah mengirimkan RREP, jika *node* menerima RREP dengan hop-count yang lebih kecil, *node* akan mengupdate informasi routing dengan jalur yang terbaik dan mengirimkannya.
- *Loop-free routes* : Algoritma menjaga agar tidak terjadi loop, dengan cara membuang jalur yang buruk dari broadcast-id yang sama.
- Topologi dinamis : Saat *node* yang berada didalam *network* bergerak atau terjadi kerusakan, topologi jaringan akan berubah, *intermediate node* yang mengetahui terjadinya kerusakan (*link breakage*) mengirimkan paket RERR.

Kekurangan AODV

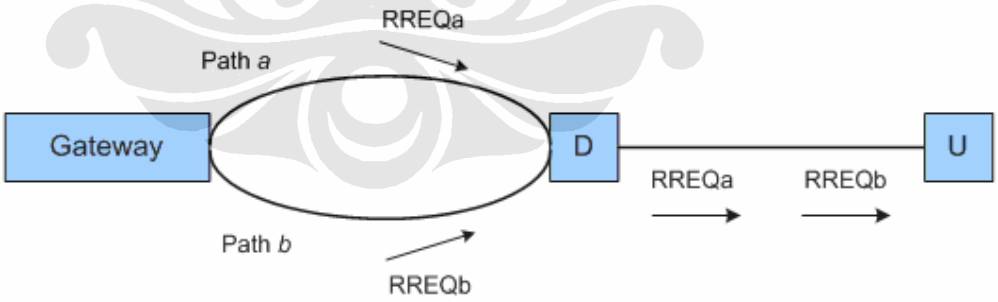
- Overhead pada bandwidth : Overhead pada bandwidth akan terjadi saat RREQ melintasi dari *node* satu ke *node* yang lain dalam proses menemukan informasi rute terbaik dan mengirimkan jalur untuk kembali, *node* yang dilewati akan membawa semua informasi dalam perjalananya.
- Informasi routing hanya dapat dipakai sekali : AODV kurang efisien dalam melakukan routing, informasi rute diperoleh berdasarkan permintaan (on-demand).
- Pencarian rute yang cukup lama, *latency* yang tinggi.
- Ukuran routing tabel yang besar.

2.6 Protokol *Routing* AODV-ST

Pada AODV-ST, *gateway* secara periodik melakukan *broadcast* RREQ untuk mulai membentuk *spanning tree*. Sebelum RREQ di *broadcast*, *gateway* menetapkan *destination-only flag* pada RREQ dan menetapkan alamat tujuan RREQ ke jaringan. Seting ini membedakan dari RREQs yang normal terhadap RREQ yang dibentuk oleh *spanning-tree*. Sebuah RREQ terdapat *metric field* yang di set *zero* oleh *gateway*. Ketika *intermediate* relay menerima RREQ, ia akan memeriksa RREQ tersebut, jika kondisinya memenuhi relay akan membentuk rute balik ke *gateway* melalui jalur terbaik. Relay tersebut dapat melakukannya karena pada RREQ terdapat *metric field*. *Field* ini selalu diupdate oleh *intermediate relay* jalur yang menunjukan karakteristik dari jalur yang telah dibentuk. Ketika *node* membentuk rute balik akan dicatat oleh *node*, *node* tersebut mengirimkan RREP secara acak kembali ke *node* asal. RREP yang dikirim secara

acak juga memiliki *metric field* yang diset *zero*. *Metric field* akan diupdate saat *intermediate* relay kembali ke *gateway*. Saat *intermediate* relay menerima RREP, *intermediate* relay akan mengirimkan terus jalur yang telah dilewati ke *node* asal. dan melakukan update ke *node* asal menggunakan metric yang terdapat pada RREP.

Pada gambar 2.10 sebuah *relay* dapat menerima dua buah *route request* untuk memilih jalur yang terbaik**.** Relay D menerima dua RREQs dari *gateway* G yang melewati dua jalur yang berbeda a dan b. Sebagai perkiraan jika RREQa mengalami sedikit *delay* dari RREQb. Saat D menerima RREQb, ia akan melakukan *broadcast* ulang karena itu adalah jalur pertama yang diketahui. Ketika RREQa yang mengalami *delay* tiba di D, maka *node* D akan melakukan *rebroadcast* ulang untuk memilih jalur yang lebih baik. Sebuah *relay* melakukan *broadcast* ulang ke *node* yang mengirim RREQ hanya jika jalur yang dibentuk oleh RREQ adalah jalur terbaik yang dapat dilalui oleh relay. *Intermediate relay* tidak akan menunggu sampai menerima semua RREQ sebelum menentukan jalur terbaik untuk di *broadcast* ulang. Ini akan mengurangi *route latency*. Ini berarti relay akan menerima duplikat RREQ dari *relay* tujuan jika duplikat dari RREQ menunjukan jalur yang lebih baik. RREQ akan di *broadcast hop-by-hop* melewati *mesh network, spanning tree* secara implisit terbentuk dari rute balik ke ke *gateway*.



Gambar 2.10 Duplikat Route Request Pada AODV-ST

AODV-ST memilik kelebihan dibanding dengan protokol AODV karena protokol AODV-ST memberikan *throughput* yang tinggi dalam bentuk *Expected Transmission Count* (ETX) dan *Expected Transmission Time* (ETT), selain itu secara proaktif menjaga *spanning tree* pada *mesh network* yang secara signifikan

mengurangi *route latency*. ETX adalah jumlah transmisi yang dibutuhkan untuk sampai ke *node* tetangganya. Untuk menghitung ETX setiap *node* secara periodik melakukan pemeriksaan untuk setiap *broadcast* yang dikirim oleh tetangganya.

Setiap *spanning tree* dibentuk demikian saat relay *node*s berada pada jalur optimal menuju *gateway* yang tepat. *Route maintenance* dijaga agar tetap minimum karena *relay* pada *spanning tree* secara periodik di *update*. Sebuah *relay* secara spesifik memilih *default gateway* yang mampu memberikan kapasitas terbesar. Untuk komunikasi relay-relay, AODV-ST menggunakan *proactive route discovery* yang ada pada AODV, secara konsep AODV-ST adalah protokol *routing hybrid*; ia menggunakan protokol *routing proactive* untuk mencari rute yang biasa digunakan pada *end-point* (*relay*-to-*gateway*), dan menggunakan protokol *routing* reaktif untuk merutekan *relay-to-relay*. Pada AODV-ST, *Expected Transmission Time* (ETT) adalah waktu yang dibutuhkan sebuah paket data untuk sampai ke-setiap tetangganya.

2.7 OpenWRT

OpenWRT adalah *thirdpary firmware* yang digunakan untuk menjalankan *wireless router* linksys seri WRT54. WRT54 adalah *wireless* router pertama yang dapat menggunakan OpenWRT karena mempunyai basis disain dari Broadcom **[15].** OpenWRT *firmware* menggunakan sistem operasi Linux yang digunakan dalam suatu *embedded device* seperti *wireless router*. Dibentuk pada akhir tahun 2003 pada awalnya openWRT hanya digunakan oleh Linksys WRT54G yang terbentuk dalam rangka mengembangkan sebuah *third-party firmware.* Dalam perkembangannya OpenWRT dapat pula digunakan untuk mendukung *wireless router* yang lain seperti ASUS, D-Link, DELL dan lain-lain.

OpenWRT hanya menyediakan *firmware* dengan paket tambahan yang memunginkan untuk merubah, menambah dan menghilangkan paket yang diinginkan. Dalam perkembangan OpenWRT sangat dipengaruhi oleh kemudahannya dalam memodifikasi fitur-fitur tambahan diluar fitur-fitur yang telah disediakan oleh pihak manufaktur agar dapat digunakan sesuai dengan keperluan tertentu dari para pengguna. Hal ini dapat dilakukan karena OpenWRT bersifat *opensource* karena dibuat berdasarkan GNU *General Public*

License/*Linux*, sehingga setiap perubahan yang dibuat oleh pihak manufaktur harus didaftarkan dan dirilis melalui lisensi GPL. Berdasarkan sifat *opensource* ini pula maka para pengguna dapat dengan bebas memodifikasi ataupun menambahkan fitur-fitur lain pada *router* sesuai dengan kebutuhan.

Dalam *firmware* bawaan dari pabrikan *wireless router* yang menggabungkan semua paket dalam satu *firmware*, maka OpenWRT dapat menyediakan konfigurasi minimal yang dibutuhkan oleh sebuah *wireless router*, namun dengan kemampuan untuk mendukung paket-paket tambahan. Untuk *wireless router* ini adalah penghematan ruang, karena paket-paket yang tidak diperlukan dapat dihilangkan. Cara umum yang digunakan untuk konfigurasi OpenWRT adalah dengan *read-only* partisi menggunakan *squashfs* atau dengan *read-write* partisi. Pada partisi kedua terdapat link ke partisi *squashfs* sebagai *root* pada file sistem **[**16**].** Ada dua cara untuk melakukan instalasi OpenWRT melalui Web GUI atau TFTP (*Trivial File Transfer Protocol*), selain itu ada dua pilihan file system dalam OpenWRT yaitu : SQUSHFS dan *Journaling Flash File System, version2* (JFFS2). Squashfs memiliki ukuran yang lebih kecil dibandingkan dengan file JFFS2 karena filenya telah dikompresi, dan cara yang paling aman untuk menginstal *firmware* Open WRT karena filenya bersifat *read-only* dan banyak direkomendasikan untuk pemula karena squashfs mampu untuk mencegah error dalam penginstalasiannya. JFFS2 adalah file sistem GNU *license*, dalam filenya tidak dilakukan kompresi sehingga membutuhkan memory yang cukup besar dan file sistemnya bersifat *read*/*wirite* dan mudah mengalami *user error* karena keterbatasan pengguna sehingga peralatan tidak dapat dipakai kembali [17].

Saat ini ada dua jenis OpenWRT yang telah dibuat yaitu :

- **WhiteRussian**

  Versi ini adalah versi pertama dari OpenWRT, lebih stabil karena telah dikembangkan paling awal. Karena itu banyak dokumentasi maupun tutorial yang tersedia untuk mendukung pemakaian OpenWRT versi ini. Versi terakhir dari whiterussian adalah White Russian 0.9 yang dirilis pada tanggal 5 Februari 2007.

- **Kamikaze**

  Versi baru dari OpenWRT, walaupun sudah stabil namun masih dalam pengembangan. Memiliki GUI yang lebih lengkap dari versi terdahulu, Dibuat berdasarkan disain berbeda dengan versi yang terdahulu sehingga dapat bekerja pada pilihan jenis *wireless router* yang lebih luas, selain itu mempunyai kernel yang lebih baru. Versi terakhir dari kamikaze adalah Kamikaze 7(09).

2.7.1 Directory Structure

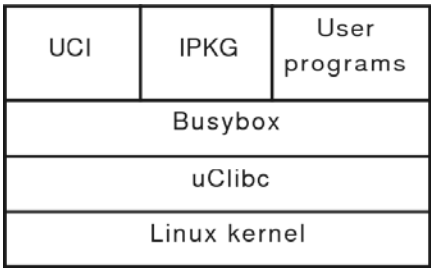
Terdapat empat directory utama pada Open WRT yaitu

- Tools
- toolchain
- package
- target

Tools dan tooolchain adalah *directory* yang biasa digunakan untuk membangun *image frimware*, yaitu *compiler* dan C *library*. Hasilnya adalah tiga *directory* baru yaitu, build_dir/host ini adalah *temporary directory* untuk membangun target, build_dir/toolchain, *directory* ini digunakan untuk membangun toolchain dengan arsitektur yang spesifik, terakhir adalah staging_dir/toolchain *directory* ini adalah hasil dari *toolchain* yang telah diinstal.

2.7.2 Software Architecture

OpenWRT menggunakan embeded linux tools seperti uClib, busybox, shell interpreter. Setiap arsitektur menggunakan *kernel* Linux berbeda yang mengijinkan *user* untuk mengembangkan. Untuk itu kita hanya butuh *recompile* uClib dan *packages* untuk mencocokan target arsitektur untuk mendapatkan program diinginkan yang berbeda dari *embeded device*. *Unified Configuration Interface* (UCI) adalah interface dari C *library* yang menyediakan hubungan konfigurasi untuk sistem. UCI digunakan oleh Open WRT untuk device yang tidak memiliki NVRAM untuk tempat meyimpan partisi. Karena UCI adalah library dari C, ini mudah diintegrasikan kedalam aplikasi yang telah ada untuk dikembangkan konfigurasi yang kompatibel dengan openWRT



Gambar 2.11 Arsitektur Software OpenWRT

OpenWRT menggunakan sistem ipkg seperti yang biasa digunakan Linux untuk mengatur paket-paket fitur tersebut. IPKG adalah sebuah *directory* tempat penyimpanan semua kontrol file ipkg. File ini selalu bernama “pkgname.type” sebagai contoh : strace.control. file ini secara otomatis ditambahkan ke package saat dieksekusi.