BAB 2
WIRELESS LOCAL AREA NETWORK

2.1 Standar IEEE 802.11 [4]

Pada awalnya, jaringan *Wireless Local Area Network* (WLAN) digunakan pada sebuah kantor perusahaan yang ingin menghubungkan penggunanya dengan medium tanpa kabel. Teknologi ini kemudian berkembang menjadi sumber daya yang sangat berharga di berbagai tempat seperti di kampus-kampus, di bandara, di rumah sakit dan lain-lain. Kemampuannya untuk menghubungkan jaringan lokal secara *wireless*, dari sembarang tempat melewati kampus atau melalui bangunan kantor, merupakan solusi yang tepat untuk menghubungkan jaringan.

Spesifikasi jaringan *Wireless Local Area Network* (WLAN) didasari pada standar IEEE 802.11 yang dikembangkan pada tahun 1997. Awalnya standar ini menggunakan transmisi data dengan kecepatan hingga 2 Mbps. Dengan berkembangannya waktu, implementasi dari standar ini semakin populer dan meluas. Penambahan ekstensi di belakang 802.11 dipergunakan untuk mengenal beberapa perbaikan dan tambahan fitur dari standar yang telah ditentukan 802.11.

Pada tahun 1999, IEEE telah meratifikasi dua standar, yaitu standar 802.11a dan 802.11b. Standar 802.11a didasarkan pada skema modulasi *Orthogonal Frequency Division Multiplexing* (OFDM) yang bekerja pada frekuensi 5 GHz dengan kecepatan transmisi data mencapai 54 Mbps. Sedangkan standar 802.11b menggunakan skema modulasi *Direct Sequence Spread Spectrum* (DSSS) dengan kecepatan transmisi data mencapai 11 Mbps pada frekuensi 2,4 Ghz.

Spesifikasi 802.11g diratifikasi pada bulan juni 2003, dimana spesifikasi ini beroperasi pada frekuensi 2,4 GHz dengan teknik modulasi yang sama dengan standar 802.11a yaitu OFDM. Standar ini memberikan kecepatan transmisi data hingga 54 Mbps.

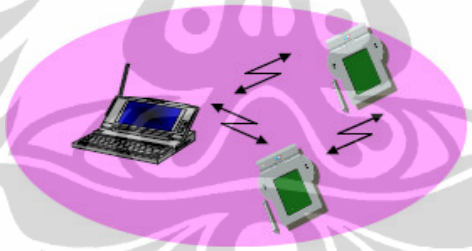
Secara paralel, beberapa standar 802.11 yang lain juga dikembangkan. Standar 802.11h ditujukan untuk memperbaiki 802.11a dengan penambahan regulasi lisensi *outdoor* dan *indoor* pada frekuensi 5 GHz di Eropa. Standar 802.11n menggunakan antena *Multiple-Input Multiple-Output* (MIMO) dan skema

modulasi adaptif OFDM. Standar ini dapat mendukung kecepatan transmisi data hingga 100 Mbps.

2.2 Arstektur IEEE 802.11 WLAN [4]

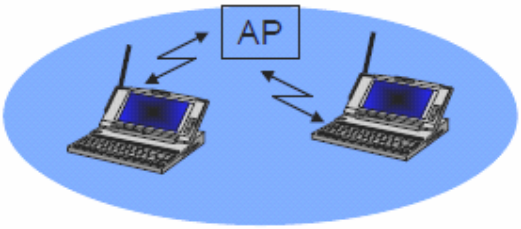
Arsitektur jaringan WLAN mengacu pada standar 802.11, yang mendukung tiga topologi dasar untuk WLAN. Ketiga topologi tersebut adalah *Independent Basic Service Set* (IBSS), *Basic Service Set* (BSS), dan *Extended Basic Service Set* (EBSS).

IBSS merupakan topologi yang paling sederhana, dimana *node-node* yang independen akan saling berkomunikasi secara *peer to peer* atau *point to point.* Topologi ini juga dikenal sebagai *ad-hoc.* Umumnya, penerapan topologi ini hanya mengkover area yang terbatas dan tidak dihubungkan ke jaringan yang besar. Topologi ini sangat mudah diterapkan dan sangat efektif untuk membangun lingkungan *wireless*-nya, seperti pada ruangan konfrensi, kelas, atau bahkan lingkungan kerja yang relatif kecil.



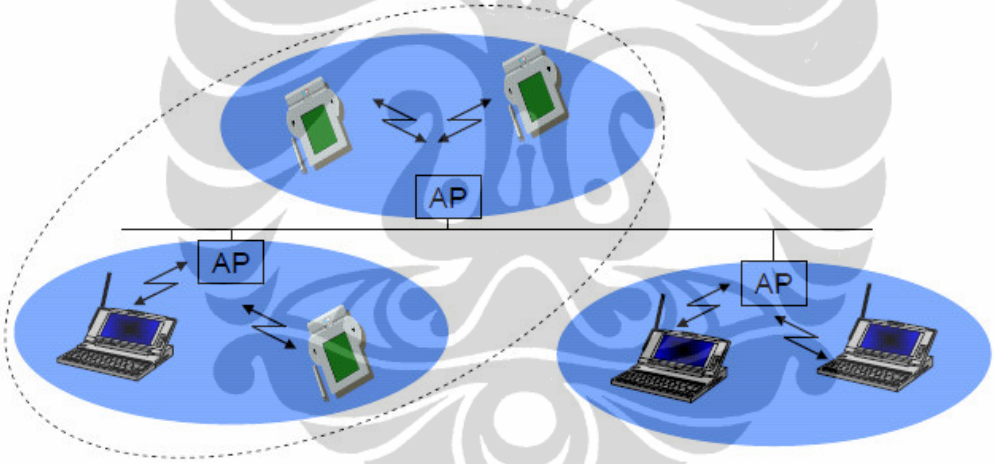
Gambar 2.1. Topologi Jaringan *Independent Basic Service Set* (IBSS)

Topologi yang lebih kompleks adalah topologi infrastruktur, atau disebut juga sebagai *Basic Service Set* (BSS). Pada topologi ini, jaringan *wireless* terdiri dari paling sedikit satu *Access Point* (AP) yang bertindak sebagai *base station* dan sekumpulan *node-node wireless*. Komunikasi antara dua *node* melewati AP. AP diperlukan untuk melakukan fungsi singkronisasi dan koordinasi, melakukan *forwarding* serta *broadcasting* paket data.



Gambar 2.2. Topologi Jaringan *Basic Service Set* (BSS)

EBSS merupakan topologi dimana beberapa AP dapat digunakan untuk membentuk daerah cakupan yang lebih luas. Topologi ini terdiri dari dua atau lebih BSS yang terkoneksi pada satu jaringan kabel. Setiap AP diatur dalam *channel* yang berlainan untuk menghindari terjadinya interferensi. Metode ini akan membentuk sel-sel seperti pada jaringan selular.



Gambar 2.3. Topologi Jaringan *Extended Basic Service Set*

2.3 Protokol MAC pada WLAN

Karena *channel* radio pada WLAN merupakan sumber yang terbatas, yang harus di*share* ke semua *user*, masing-masing stasiun harus bersaing dengan stasiun yang lainnya untuk mendapatkan akses. Hal ini diperlukan, karena hanya satu transmisi yang dapat diijinkan pada *channel* WLAN pada sembarang waktu. Protokol MAC melakukan fungsi ini yang menentukan kapan suatu *node* diijinkan untuk transmit pada medium. Protokol MAC memiliki pengaruh yang sangat besar terhadap kapasitas sistem secara keseluruhan. Selain itu, parameter-

parameter MAC merupakan faktor utama dalam menilai tingkat kinerja yang dicapai berdasarkan skenario yang diberikan.

Pada subbab berikut ini akan dijelaskan mekanisme protokol MAC standar 802.11, yaitu *Distributed Coordination Function* (DCF) dan *Point Coordination Function* (PCF). Selanjutnya juga akan dilihat protokol MAC yang memberikan dukungan QoS 802.11e. Standar perbaikan ini terdiri dari *Enhanced Distributed Coordination Access* (EDCA) dan HCF *Controlled Channel Access* (HCCA).

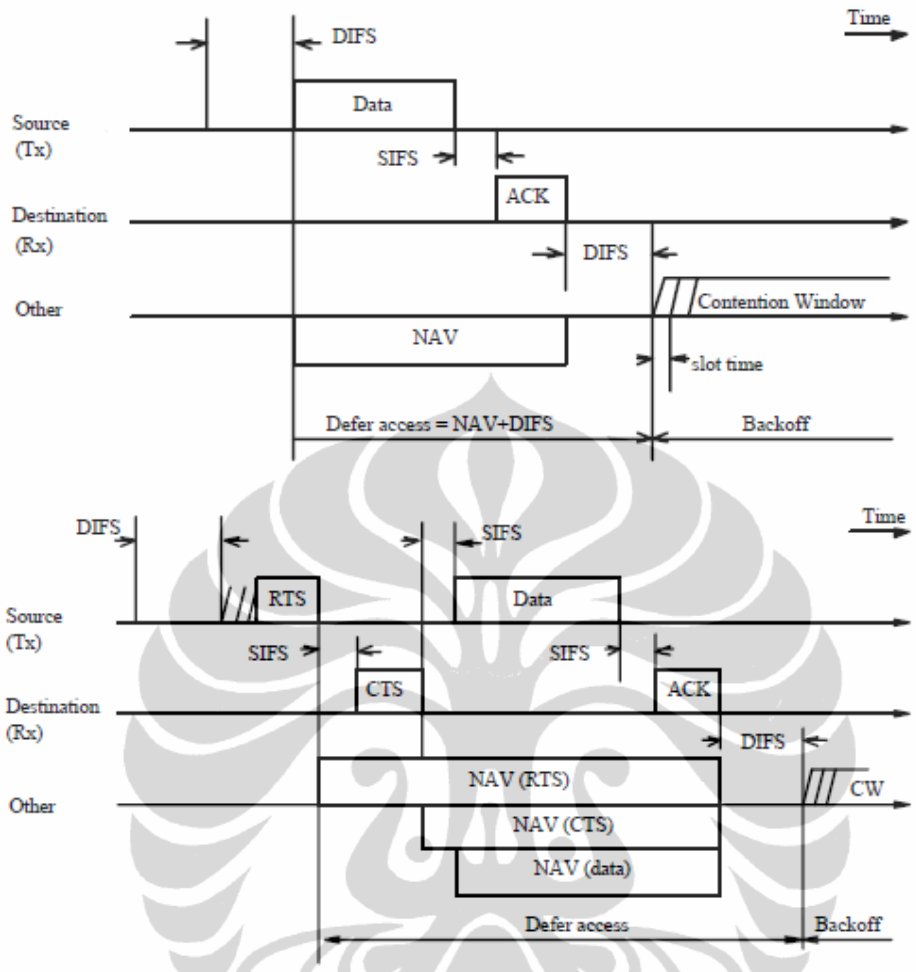
2.3.1 Standar MAC pada IEEE 802.11

Protokol MAC jaringan WLAN pada awalnya mengacu pada standar 802.11. Protokol MAC pada standar ini minim dukungan QoS yang hanya dirancang untuk memberikan layanan *best effort.*

Ada dua protokol MAC yang didasari pada standar 802.11 ini. Yang pertama adalah *Distributed Coordination Function* (DCF) yang beroperasi pada *Contention Period.* Hal ini menyebabkan semua *node* untuk memperebutkan kanal untuk melakukan pengiriman. Sedangkan yang kedua adalah *Point Coordination Function* (PCF) dimana medium dapat berpindah antara *Contention Period* dan *Contention-Free Period.* Subbab berikut ini akan lebih menjelaskan dua protokol MAC tersebut.

2.3.1.1 *Disributed Coordination Function* (DCF) [1, 3]

DCF didasarkan pada skema *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). Karena perbedaan yang signifikan antara level daya yang ditransmisikan dengan yang diterima, *collision detection* tidak dapat diterapkan. Secara aktual, ada dua mekanisme *sensing carrier* yang digunakan: PHY *carrier sensing* pada *interface* udara dan *virtual carrier sensing* pada lapisan MAC.



Gambar 2.4. Mekanisme Akses DCF CSMA/CA (Atas) Dan Skema RTS/CTS (Bawah) [1]

PHY *carrier sensing* mendeteksi keberadaan stasiun lain dengan menganalisa semua paket yang diterima dari stasiun lain. *Virtual carrier sensing* yang merupakan tambahan digunakan oleh stasiun untuk menginformasikan ke semua stasiun yang lain di dalam BSS atau IBSS, berapa lama *channel* akan diduduki untuk pengiriman *frame*-nya. Untuk menghindari skenario ini, pengirim dapat mengeset *field* durasi MAC *header* pada *frame-frame* data, atau pada *frame-frame* kontrol *Request To Send* (RTS) dan *Clear To Send (CTS).* Kemudian, stasiun-stasiun yang lain akan memperbaharui *timer* lokal dari *Network Allocation Vector*-nya (NAV) untuk menghitung durasi ini. Seperti ditunjukkan pada gambar 2.4, bila sebuah paket diterima pada antrian yang kosong dan bila medium *idle* untuk *interval* waktu yang lebih panjang dari *Distributed Interframe Space*

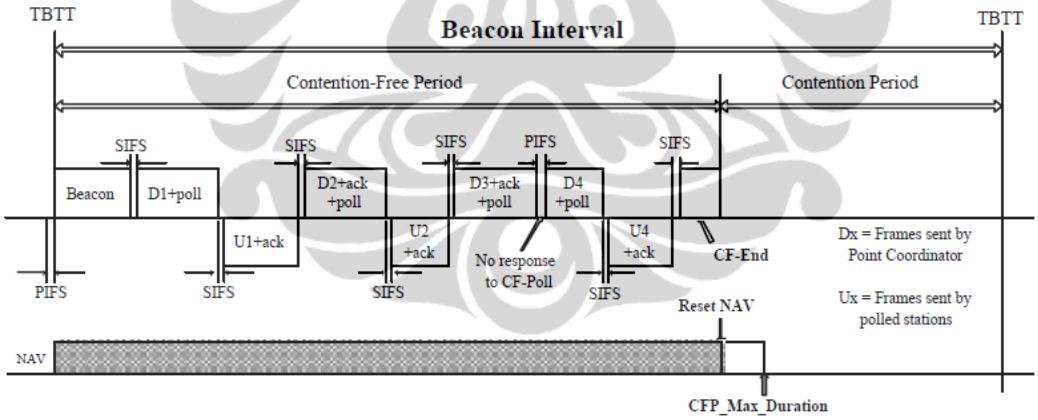
(DIFS), stasiun sumber dapat mengirim paketnya segera. Sementara itu, stasiun yang lain menunda pengirimannya dengan menggeser NAV-nya, dan memulai proses *backoff.* Lebih tepatnya, stasiun-stasiun menghitung *interval* waktu random, yang disebut *backoff timer,* dipilih dari *Contention Window* (CW): *backoff timer* = *rand* [0,*CW*] × *slot time*, dmana $CW_{min} < CW < CW_{max}$ dan *slot time* tergantung pada jenis PHY *layer*. Parameter *backoff timer* menurun hanya bila medium *idle*, konstan bila stasiun yang lain melakukan pengiriman. Setiap saat medium menjadi *idle*, stasiun menunggu selama DFS dan secara kontinyu menurunkan *backoff timer*-nya. Bila *backoff timer* telah habis, stasiun diijinkan untuk mengakses medium. Tabrakan terjadi bila paling sedikit ada dua stasiun memulai pengiriman secara serentak. Untuk tujuan ini, *positive acknowledgement* (ACK) digunakan untuk memberitahukan pengiriman bahwa *frame* yang dikirimkan telah berhasil diterima, seperti dapat dilihat pada gambar 2.4. Bila ACK tidak diterima, pengirim mengasumsikan bahwa telah terjadi tabrakan, dan pengirim mengatur kembali pengiriman ulang dengan memasuki proses *backoff* kembali. Untuk mengurangi kemungkinan tabrakan, setelah pengiriman yang tidak berhasil terjadi, nilai *CW* diduakalikan hingga nilai CW_{max}nya. Setelah pengiriman yang berhasil, nilai *CW* di-*reset* ke nilai minimumnya CW_{min}.

2.3.1.2 *Point Coordination Fuction* (PCF) [1, 3]

Akses berdasarkan prioritas dapat juga digunakan untuk mengakses medium. Sebagai contoh, PCF merupakan mekanisme akses yang menerapkan skema akses *polling-based contention-free* dan hanya digunakan pada topologi jaringan infrastruktur. Tidak seperti DCF, implementasi PCF tidaklah wajib. Alasannya adalah bahwa penerapan hardware PCF sangatlah kompleks pada saat standar ini dibuat. PCF menggunakan skema *polling* terpusat, yang memakai AP sebagai *Point Coordination* (PC). Bila BSS diset dengan PCF-*enabled*, waktu akses kanal dibagi ke dalam *interval* periodik yang disebut *beacon interval*, seperti terlihat pada gambar 2.5. *Beacon interval* terdiri dari *Contention-Free Period* (CFP) dan *Contention Period* (CP). Selama CFP, PC menjaga daftar stasiun yang teregister dan mem-*poll* mereka sesuai dengan daftar tersebut. Ketika stasiun di-*poll*, ia mulai mengirim *frame* data, dimana ukuran masing-masing

frame data dibatasi oleh MAC *Service Data Unit* maksimum. Asumsi bahwa PHY *rate* dari tiap stasiun tetap, maksimum durasi CFP untuk semua stasiun, yang disebut *CFP_max_duration,* kemudian ditentukan oleh PC. Sebaliknya, kemampuan *link-adaptation* membuat waktu transmisi dari sebuah *frame* bervariasi dan dapat mengakibatkan besarnya *delay jitter,* yang menurunkan kinerja QoS dari PCF.

Waktu yang digunakan oleh PC untuk membangkitkan *frame-frame beacon* disebut *Target Beacon Transmission Time* (TBTT). Untuk memberikan PCF dengan prioritas yang lebih tinggi untuk akses dibanding DCF dalam *interval beacon*, PC menunggu selama *interframe space* yang lebih pendek dari DIFS (disebut PIFS), sebelum memulai PCF. Tetapi PCF tidak diijinkan untuk melakukan pengiriman *frame* dalam PCF. Kemudian, semua stasiun yang lain mengatur NAV-nya ke nilai CFP_max_duration, atau durasi sisa dari CFP dalam kasus *beacon* yang tertunda. Urutan pengaksesan medium selama PCF ditunjukkan pada gambar 2.5.



Gambar 2.5. PCF Dan DCF Yang Bergantian [3]

Bila PCF mendapatkan akses ke medium *wireless*, SIFS (*Short Interframe Space*) *timing* digunakan untuk pertukaran *frame* selama CFP kecuali bila stasiun yang di-*poll* tidak merespon PC pada periode PIFS. Bila PC mem-*poll* stasiun, ia dapat melakukan *piggyback frame-frame* data ke stasiun bersama dengan CF-*Poll*, kemudian stasiun mengirim kembali *frame* data *piggybacked* dengan ACK setelah *interval* SIFS. Bila PC mem-*poll* stasiun berikutnya, ia dapat *piggyback* tidak

hanya *frame* data ke tujuan, tetapi juga ACK yang berhubungan dengan pentrasmisian yang berhasil sebelumnya. Stasiun-stasiun yang diam dapat dipindahkan dari daftar *polling* setelah beberapa periode dan dapat di-*poll* kembali pada awal CFP berikutnya. Ingatlah bahwa pada sembarang waktu, PC dapat memutuskan untuk mengakhiri CFP dengan mengirimkan *frame* CF-*End*. Biasanya, PCF menggunakan *Round-Robin scheduler* untuk mem*poll* masing-masing stasiun secara berurutan dalam urutan daftar polling, tetapi mekanisme polling yang berdasarkan prioritas dapat juga digunakan bila tingkat QoS yang berbeda diminta stasiun-stasiun yang berbeda.

2.3.2 Standar Perbaikan QoS MAC pada IEEE 802.11e

Standar 802.11 awalnya dikembangkan untuk mendukung aplikasi *best effort*. Pada saat ini, aplikasi *real time* seperti audio dan video, yang memiliki batasan kinerja yang lebih baik, semakin banyak digunakan. Oleh karena itu, untuk mendukung aplikasi tersebut agar dapat digunakan pada WLAN dengan tingkat QoS yang dapat diterima, maka dukungan pembedaan dan prioritas trafik diperlukan. Hal inilah yang merupakan faktor pemicu munculnya standar 802.11e yang diperlukan untuk memperbaiki kinerja jaringan WLAN.

Pada standar IEEE 802.11e ini, terjadi perubahan penamaan pada stasiun atau *node* yang menjadi cakupan jaringan serta AP yang menjadi koordinatornya. Setiap stasiun atau *node* pada jaringan IEEE 802.11e disebut sebagai QoS-*enabled Station* (QSTA) sedangkan AP-nya disebut QoS-*enabled Access Point* (QAP). Hal ini dikarenakan pada IEEE 802.11e, setiap stasiunnya memiliki kemampuan yang berbeda dibanding pada IEEE 802.11.

Standar 802.11e memberikan *service differentiation* yang diperlukan dengan mengumpulkan tingkatan prioritas dari masing-masing paket. Paket-paket dengan prioritas yang lebih tinggi mendapatkan hak akses yang lebih tinggi ke medium wireless. Jadi skema ini memberikan *resource* ke paket berdasarkan tingkat kinerja yang diperlukannya.

Pada subbab berikut ini akan dijelaskan dua protokol MAC yang memberikan dukungan QoS pada standar 802.11e, yaitu *Enhanced Distributed Coordination Access* (EDCA) dan HCF *Controlled Channel Access* (HCCA).

2.3.2.1 *Enhanced Distributed Coordination Access* (EDCA) [1, 4]

EDCA dirancang untuk memberikan prioritas QoS dengan memperbaiki DCF. Sebelum masuk ke MAC *layer*, tiap-tiap paket data yang diterima dari *layer* yang lebih tinggi, ditandakan ke nilai prioritas user tertentu. Bagaimana untuk menandakan nilai prioritas ini ke tiap-tiap paket merupakan masalah penerapannya. Pada MAC *layer*, EDCA memperkenalkan 4 antrian FIFO yang berbeda, yang disebut *Access Categories* (AC). Seperti ditentukan pada draft 802.11e, masing-masing paket data dari *layer* yang lebih tinggi dengan nilai prioritas user tertentu harus dipetakan ke AC yang berhubungan dengan menggunakan tabel pemetaan. Nilai prioritas *user* ditentukan dalam spesifikasi *bridge* 802.11d.

Tabel 2.1. Pemetaan Antara Prioritas *User* Dan *Access Categories* (AC) [4]

Prioritas *user*	AC	Penandaan
1	AC_BK	*Background*
2	AC_BK	*Background*
0	AC_BE	*Best Effort*
3	AC_BE	*Best Effort*
4	AC_VI	Video
5	AC_VI	Video
6	AC_VO	*Voice*
7	AC_VO	*Voice*

Seperti ditunjukkan pada Tabel 2.1, jenis aplikasi yang berbeda seperti trafik *background*, trafik *best-effort*, trafik video dan audio dapat dipetakan ke antrian AC yang berbeda (contoh, AC_BK, AC_BE, AC_VI, AC_VO). Setiap AC memiliki entitas DCF sendiri dan tiap-tiap entitas memiliki parameter *contention*-nya (CW_{max}[AC], CW_{min}[AC], AIFS[AC], dan TXOPLimit[AC]) masing-masing yang diberitahukan oleh QAP secara periodik melalui *frame-frame beacon*. Pada dasarnya, nilai yang paling kecil dari CW_{max}[AC], CW_{min}[AC], AIFS[AC], dan TXOPLimit[AC], *delay* akses kanalnya yang lebih tinggi untuk mengakses

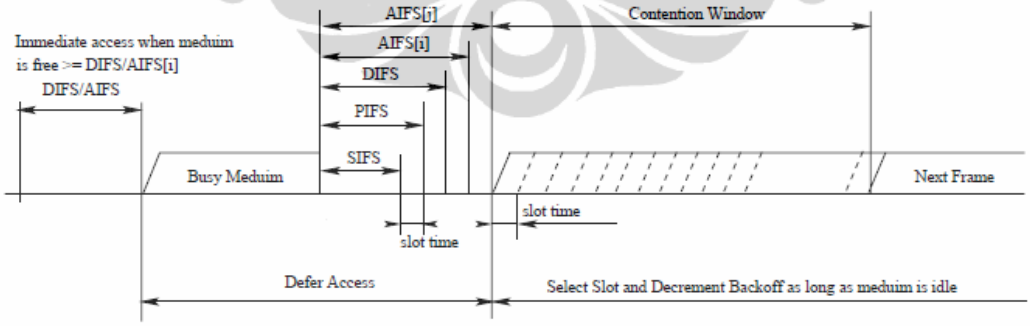
medium. Berbeda dengan skema-skema yang kebanyakan yang hanya menggunakan DIFS dan PIFS untuk pembedaan, EDCA memperkenalkan jenis baru IFS, yang disebut *Arbitrary* IFS (AIFS). Masing-masing AIFS adalah *interval* IFS dengan panjang yang berubah-ubah dan ditentukan dengan persamaan

$$AIFS[AC] = SIFS + AIFSN[AC] \times slot\ time \quad (2.1)$$

Dimana AIFSN[AC], disebut *Arbitrary Inter Frame Space Number*, yang menunjukkan jumlah *slot time* untuk sebuah AC. Sebagai contoh, DCF menggunakan AIFSN = 2 untuk menghitung DIFS (dalam contoh, DIFS = SIFS + 2 × *slot time*).

Tabel 2.2. Parameter Standar EDCA [4]

AC	CW_{min}	CW_{max}	AIFSN	TXOPLimit (802.11b)	TXOPLimit (802.11a/g)
BK	CW_{min}	CW_{max}	7	0	0
BE	CW_{min}	CW_{max}	3	0	0
VI	$(CW_{min}+1)/2-1$	CW_{min}	2	6.016ms	3.008ms
VO	$(CW_{min}+1)/4-1$	$(CW_{min}+1)/2-1$	2	3.008ms	1.504ms



Gambar 2.6. Hubungan *Interframe Space* (IFS) IEEE 802.11e [1]

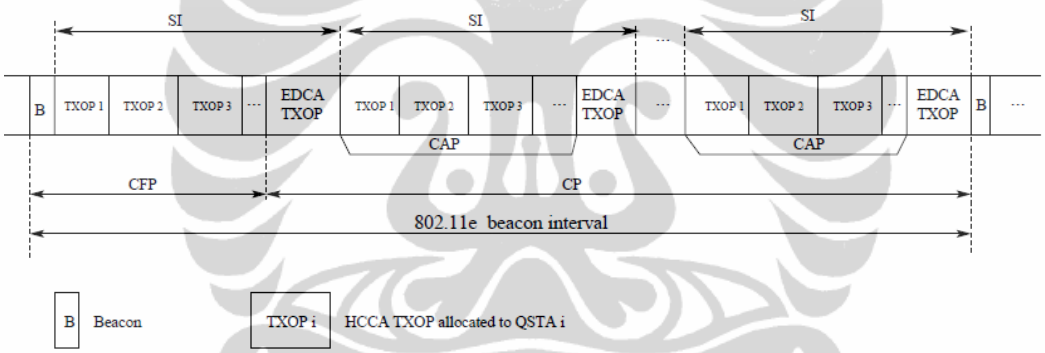
Parameter EDCA standar yang digunakan QSTA diperlihatkan pada Tabel 2.2. Setelah menunggu *interval* waktu *idle* AIFS[AC], masing-masing AC harus menunggu untuk waktu *backoff random* ($CW_{min}[AC] \leq$ *backoff time* $\leq CW_{max}[AC]$). Tujuan dari penggunaan parameter kontensi yang berbeda-beda

untuk antrian yang berbeda-beda adalah untuk trafik *low-priority* waktu *backoff* yang lebih panjang dari trafik *high-priority*. Trafik *high-priority* dapat akses ke medium lebih awal dari trafik *low-priority*. Masalah yang potensial adalah bahwa waktu *backoff* dari AC yang berbeda-beda dapat tumpang tindih dan mereduksi efek *service differentiation*. Selain itu, pada EDCA, waktu *backoff* dari AC yang berbeda-beda dalam satu QSTA merupakan nilai random dan dapat mencapai nol pada waktu yang sama, jadi menyebabkan tabrakan internal. Untuk menghindari tabrakan internal ini, EDCA memperkenalkan penjadwalan di setiap QSTA yang memungkinkan hanya AC dengan prioritas yang lebih tinggi dapat mengirimkan paket. Akibatnya, EDCA dapat mendukung QoS terprioritas untuk aplikasi multimedia.

2.3.2.2 *HCF Controlled Channel Access* (HCCA) [1, 3]

Untuk memberikan batasan dan dukungan parameter QoS tanpa memandang kondisi trafik, mekanisme HCF *Controlled Channel Access* (HCCA) diusulkan oleh 802.11e *working group*. HCCA menggunakan mekanisme *poll-and-response* yang sama dengan PCF, tetapi ada banyak perbedaan atara kedua mekanisme tersebut. Sebagai contoh, HCCA lebih fleksibel dibanding PCF, dalam hal, QAP dapat menjalankan HCCA baik pada CFP dan CP sedangkan PCF hanya diijinkan pada CFP. Kemudian, HCCA memecahkan tiga masalah yang muncul pada PCF. Yang pertama adalah adanya *link* langsung antara stasiun *peer* diperbolehkan pada 802.11e, dimana stasiun-stasiun dapat berkomunikasi dengan yang lain tanpa melalui AP. Yang kedua adalah bahwa QSTA 802.11e tidak diijinkan untuk mengirimkan paket bila pentransmisian *frame* tidak dapat diselesaikan sebelum *beacon* berikutnya, yang menyelesaikan masalah *delay beacon* pada PCF. Dan yang terakhir, TXOPLimit digunakan untuk membatasi waktu pengiriman dari stasiun yang di-*poll*. Gambar 2.7 memperlihatkan contoh *interval beacon* 802.11e (durasi antara dua *beacon* yang berurutan), yang tersusun dari mode wajib CP dan mode tambahan CFP. Selama CP, QAP diijinkan untuk memulai beberapa *burst contention-free*, yang disebut *Controlled Access Period* (CAP), pada sembarang waktu setelah mendeteksi kanal *idle* selama *interval* waktu PIFS. Seperti ditunjukkan pada Gambar 2.6, PIFS lebih pendek dibanding

DIFS dan AIFS, yang memberikan QAP probabilitas yang lebih besar untuk memulai HCCA pada sembarang waktu selama CP dibandingkan QSTA yang lain. HCCA lebih fleksibel dibanding PCF karena PCF harus terjadi secara periodik setelah *frame beacon*, sedangkan QAP dapat memulai HCCA kapanpun ia mau. Walaupun PCF masih diijinkan pada 802.11e, fleksibilitas HCCA membuat PCF tidak berguna. Jadi, PCF merupakan pilihan pada draft 802.11e. Setelah periode tambahan CFP, mekanisme EDCA dan HCCA yang digunakan dalam durasi CAP, bergantian dalam *interval beacon*. Meskipun HCCA dapat memberikan dukungan QoS yang lebih baik dari EDCA, namun EDCA masih wajib dalam 802.11e untuk mendukung pertukaran QoS yang dipersyaratkan antara QSTA dan QAP. Untuk tujuan ini, durasi maksimum HCCA pada *interval beacon* 802.11e dibatasi bervariasi, $T_{CAPLimit}$.



Gambar 2.7. Contoh *Interval Beacon* Yang Digunakan Pada Algoritma Penjadwalan HCF [3]

Algoritma penjadwalan yang sederhana diajukan sebagai rancangan acuan dalam spesifikasi 802.11e, yang memberikan dukungan QoS terparameter didasarkan pada kontrak antara QAP dan QSTA yang berhubungan. Sebelum pengiriman data, deretan trafik harus pertama-tama ditentukan dan masing-masing QSTA diijinkan untuk memiliki tidak lebih dari delapan deretan trafik dengan prioritas yang berbeda-beda. Ingatlah bahwa deretan trafik dan AC dipisahkan dan menggunakan antrian MAC yang berbeda-beda. Untuk mensetup koneksi deretan trafik, QSTA harus mengirim *frame* permintaan QoS yang mengandung spesifikasi trafik yang berhubungan (TSPEC) ke QAP. TSPEC menggambarkan

parameter QoS yang diperlukan dari deretan trafik seperti *mean data rate*, ukuran maksimum MAC *Service Data Unit* (MSDU), batasan *delay* dan maksimum *Required Service Internal* (RSI). Maksimum RSI mengacu kepada durasi waktu maksimum antara awal TXOP yang berhasil yang dapat ditolerir oleh aplikasi. Pada dasarnya, ada hubungan antara maksimum RSI dan batasan *delay* ditentukan oleh QSTA, penjadwalan sederhana HCF 802.11e hanya menggunakan maksimum RSI untuk menghitung penjadwalan TXOP. Algoritma penjadwalan HCF 802.11e yang sederhana dapat diringkas sebagai berikut:

Saat menerima semua permintaan QoS, penjadwalan QAP pertama-tama menentukan nilai minimum dari semua RSI maksimum yang diperlukan oleh deretan trafik yang berbeda-beda. Kedua, ia kemudian memilih nilai *submultiple* terbesar dari durasi *interval beacon* sebagai *Service Internal* (SI) yang dipilih, yang lebih kecil dibanding minimum dari semua RSI yang maksimum. Ketiga, *interval beacon* 802.11e dibagi-bagi ke dalam beberapa SI dan QSTA di-*poll* secara berurutan selama masing-masing SI yang dipilih. SI yang dipilih mengacu kepada waktu antara awal TXOP yang berhasil dialokasikan ke QSTA, yang sama untuk semua stasiun. Segera setelah SI ditentukan, penjadwalan QAP menghitung alokasi nilai TXOP yang berbeda-beda ke deretan trafik untuk QSTA yang berbeda-beda pula, yaitu TXOP1, TXOP2 dan seterusnya, seperti ditunjukkan pada Gambar 2.7. Bila permintaan *mean data rate* aplikasi dari deretan trafik j pada QSTA i adalah $\overline{\rho}_{ij}$ dan ukuran MSCU nominal untuk antrian ini adalah M_{ij} maka jumlah paket yang tiba pada deretan trafik selama SI yang dipilih dapat dihitung sebagai berikut:

$$N_{ij} = \left\lceil \frac{\overline{\rho}_{ij} SI}{M_{ij}} \right\rceil \tag{2.2}$$

Jadi penjadwalan QAP menghitung alokasi TXOP, T_{ij} untuk deretan trafik j pada QSTA I sebagai berikut:

$$T_{ij} = \max\left(\frac{N_{ij} M_{ij}}{R} + O, \frac{M_{\max}}{R} + O \right) \tag{2.3}$$

Dimana R adalah laju transmisi PHY layer dan M_{max} adalah ukuran maksimum MSDU. O mendacu kepada *overhead* transmisi dikarenakan *header frame*, IFS,

ACK dan *frame-frame poll* PHY/MAC *layer*. O merupakan unit waktu dan dihitung dengan 2SFIS + TACK.

Keempat, penjadwalan QAP menjumlahkan semua nilai TXOP dan deretan trafik yang berbeda-beda pada QSTA i sebagai:

$$TXOP_i = \sum_{j=1}^{J_i} T_{ij} \tag{2.4}$$

Dimana J_i adalah jumlah deretan trafik yang aktif pada QSTA i. kemudian, penjadwalan QAP mengalokasikan *interval* waktu $TXOP_i$ ke QSTA i dan mengijinkan QSTA untuk mengirimkan banyak *frame* selama *interval* waktu ini. Pada cara ini, penjadwalan QAP diharuskan untuk mengalokasikan TXOP yang berhubungan untuk mengirimkan semua *frame-frame* yang tiba selama SI yang dipilih. Jadi, penjadwalan QAP diharapkan dapat mengkontrol *delay*.

Algoritma *Admission Control* juga dianjurkan dalam penjadwalan HCF yang sederhana. Dengan menggunakan algoritma penjadwalan HCF yang sederhana. Dengan menggunakan algoritma penjadwalan di atas, bagian keseluruhan dari waktu pengiriman yang direservasi untuk HCCA dari semua QSTA K dalam *interval beacon* 802.11e dapat dihitung sebagai: $\sum_{i=1}^{K} \frac{TXOP_i}{SI}$.

Untuk memutuskan apakah ada atau tidak permintaan baru dari aliran trafik baru yang dapat diterima dalam HCCA, penjadwalan QAP hanya perlu mengecek jika permintaan yang baru dari $TXOP_{K+1}$ ditambah semua alokasi TXOP saat ini lebih rendah atau sama dengan bagian maksimum dari waktu dapat digunakan HCCA:

$$\frac{TXOP_{K+1}}{SI} + \sum_{i=1}^{K} \frac{TXOP_i}{SI} \le \frac{T_{CAPLimit}}{T_{Beacon}} \tag{2.5}$$

Dimana $T_{CAPLimit}$ adalah batasan durasi maksimum dari HCCA dan T_{Beacon} menunjukkan panjang *interval beacon*.

BAB 3
SIMULASI IEEE 802.11E HCCA DENGAN NS-2

Ada banyak teknik yang dilakukan untuk mengevaluasi kinerja suatu sistem. Salah satu teknik tersebut adalah dengan simulasi. Teknik simulasi sering digunakan untuk menguji sistem yang diusulkan dalam waktu yang singkat dan dengan biaya yang cukup rendah. Karena alasan inilah, simulator menjadi *tool* yang paling penting bagi para peneliti untuk menginvestigasi teknologi-teknologi baru.

Tujuan dari penelitian ini adalah untuk mengevaluasi kinerja MAC protokol 802.11e HCCA pada *Wireless Local Area Network*. Untuk dapat menganalisis MAC protokol tersebut, teknik simulasi digunakan.

Ada banyak *simulator* yang dapat dipertimbangkan untuk mengevaluasi kinerja MAC protokol pada WLAN. Namun pada penelitian ini, penulis menggunakan NS-2. Beberapa pertimbangannya adalah NS-2 telah digunakan secara luas dan para peneliti di dunia telah memberikan kontribusi terhadap pengembangannya. Ini berarti bahwa *simulator* ini cukup lengkap dan memiliki banyak ekstensi untuk aplikasi-aplikasi, protokol-protokol dan model tafik yang berbeda-beda.

Bagian pertama dari bab ini membahas *simulator* yang digunakan. Penerapan protokol MAC IEEE 802.11e HCCA dan protokol *routing* pada NS-2 dijelaskan secara rinci pada bagian ini. Pada bagian selanjutnya dibahas parameter-parameter yang digunakan pada simulasi, model trafik, skenario yang diberikan, juga masalah metrik kinerja yang dipakai.

3.1 *Network Simulator* [5, 9]

NS-2 merupakan *open source simulator*, yang dirancang untuk mensimulasikan *Local Area Network* maupun *Wide Area Network*. Versi pertama dari *Network Simulator* (NS) dirilis pada tahun 1995 dan merupakan varian dari *Realistic and Large* (REAL) *Simulator*, yang ditulis oleh Keshev pada tahun 1988. REAL itu sendiri dikembangkan dari *Network Tesbed* (NEST). NS pertama-tama dikembangkan sebagai bagian projek *Vitual Inter-Network Test-bed* (VINT) dan versi keduanya, NS-2 dirilis pada tahun 1996. Namun saat ini, pengembangan

NS-2 didukung melalui *Defense Advanced Research Projects Agency* (DARPA) oleh projek *Simulation Augmented by Measurement and Analysis for Network* (SAMAN) dan melalui *National Science Foundation* (NSF) oleh projek *Collaborative Simulation for Education and Research* (CONSER), yang kedua-duanya berkolaborasi dengan peneliti yang lain termasuk ICSI *Center for Internet Research* (ICIR) dan *Carnegie Mellon University* (CMU).

NS dibangun dengan menggunakan dua bahasa pemrograman: *simulator* yang berorientasi objek, yang ditulis dala bahasa pemrograman C++ dan OTcl (ekstensi yang berorientasi objek dari Tcl) *interpreter*, yang digunakan untuk menjalankan *script* perintah-perintah *user*.

NS memiliki *library* jaringan dan objek-objek protokol yang sangat banyak. Ada dua kelas hirarki: hirarki *compiler* C++ dan hirarki OTcl *interpreter*, yang keduanya saling berhubungan. Hirarki *compiler* C++ memungkinkan kita untuk mendapatkan efisiensi di dalam simulasi dan waktu eksekusi yang lebih cepat, meskipun simulasi melibatkan jumlah paket dan sumber data dalam jumlah yang besar. Sedangkan dengan *script* OTcl yang diberikan oleh *user*, kita dapat mendefinisikan topologi jaringan tertentu, protokol-protokol dan aplikasi-aplikasi tertentu yang ingin kita simulasikan (yang karakteristiknya sudah didefinisikan dalam hirarki *compiler*) dan bentuk keluaran yang ingin kita dapatkan dari *simulator*. OTcl dapat membuat penggunaan objek-objek yang di-*compiler* di dalam C++ melalui OTcl *linkage* yang membuat kesesuaian objek OTcl untuk tiap C++.

NS merupakan *discrete event simulator*, dimana kenaikan waktu tergantung pewaktuan *event* yang datur oleh *scheduler*. *Event* adalah suatu objek di dalam hirarki C++ dengan ID yang unik, waktu penjadwalan dan *pointer* ke objek yang menangani *event*. *Scheduler* menjaga struktur data yang berurutan dengan *event-event* yang dijalankan dan menyalakannya satu per satu.

3.1.1 Implementasi IEEE 802.11e HCCA pada NS-2 [2]

Distribusi standar dari NS-2 tidak mendukung protokol MAC IEEE 802.11e HCCA. Tambahan dan perbaikan untuk mendukung protokol tersebut

diberikan oleh *Computer Networking Group* dari *University of Pisa* [2]. Tambahan dan perbaikan tersebut terdiri dari tiga modul, yaitu:

- *Classifier*, digunakan untuk menandakan paket-paket yang datang dari *transport agent* dengan *Traffic Stream Identifier* (TID) yang berhubungan.
- HCCA *scheduler*, yang digunakan pada QoS *Access Point* (QAP) dan QoS *Station* (QSTA) untuk mengatur paket-paket HCCA *downstream* sesuai parameter-parameter QoS dari deretan trafiknya.
- Modul MAC yang dimodifikasi, yang memperbaiki MAC *class* standar IEEE 802.11.

3.1.1.1 Modul *Classifier*

Fungsi dari modul *classifier* adalah untuk menandakan paket-paket secara benar yang dimiliki oleh deretan trafik yang dibentuk dengan *Traffic Identifier* (TID). Hanya paket-paket dari *link layer* ke MAC yang ditandakan, karena paket-paket *uplink* hanya dilewatkan ke *layer* yang lebih tinggi tanpa mekanisme penjadwalan.

Masing-masing stasiun menjalankan *classifier* yang terpisah, yang dapat dikategorikan ke dalam dua jenis:

- *Classifier* untuk QSTA: aturan penandaan didasarkan pada kumpulan aturan tertentu dari terminal.
- *Classifier* untuk QAP: aturan penandaan didasarkan pada aturan di atas dan pengidentifikasian dari QSTA tujuan.

Modul *scheduler* MAC dan HCCA dapat mencari TID dari sembarang paket.

3.1.1.2 Modul MAC

Pada modul ini menggambarkan struktur data utama, fungsi-fungsi dan *event-event* pada modul MAC.

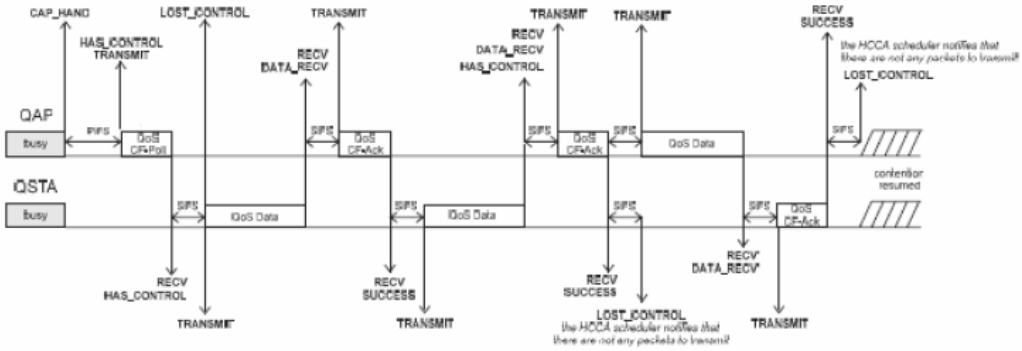
Sedangkan beberapa *event* yang diturunkan yang menggambarkan kondisi *state* MAC:

- *HCCA_HAS_CONTROL. Event* ini menyatakan bahwa stasiun memiliki kontrol terhadap medium. QSTA membangkitkan *event* ini bila ia di-*poll* oleh QAP. QAP membangkitkan *event* ini bila ia men-*sense* medium *idle*

untuk periode waktu yang lebih besar dari PIFS atau bila medium menerima *frame* terakhir dari QSTA selama *burts* TXOP.

- *HCCA_LOST_CONTROL. Event* ini menyatakan stasiun bahwa ia tidak memiliki kontrol terhadap medium lagi. Ia dibangkitkan bila HCCA *scheduler* memiliki paket yang akan dikirimkan. Juga, QAP membangkitkan *event* ini bila QSTA merespon *frame polling* dan QSTA membangkitkan *event* ini bila QAP secara benar menjawab *frame* terakhir dari TXOP *burst*.
- *HCCA_DATA_RECV*. *Event* ini menyatakan stasiun bahwa sebuah *frame* yang membawa data yang dialamatkan ke stasiun ini telah diterima secara benar.
- *HCCA_RECV*. *Event* ini menyatakan bahwa sembarang jenis *frame* yang dialamatkan ke stasiun ini telah diterima secara benar.
- *HCCA_SUCCESS*. *Event* ini menyatakan stasiun bahwa *frame* data *downlink* telah dijawab oleh stasiun penerima.
- *HCCA_TRANSMIT*. *Event* ini menyatakan stasiun bahwa *frame downlink* telah dikirim.
- *HCCA_TX_END*. *Event* ini dibangkitkan oleh QAP bila TXOP yang dijamin ke QSTA telah berakhir.
- *HCCA_CAP_HAND*. *Event* ini dibangkitkan oleh QAP bila inilah waktunya untuk memulai CAP yang baru, sesuai permintaan HCCA *scheduler*.

Gambar 3.1 menunjukkan *event* di atas dalam kasus pertukaran *frame*: pengiriman dua *frame* data *uplink*, diikuti dengan pengiriman *frame* data *downlink*, diasumsikan bahwa tidak ada tabrakan atau *frame* yang *corrupt* karena kanal yang jelek terjadi.



Gambar 3.1. *Event-Event* Selama Deretan Pertukaran *Frame*

3.1.1.3 Modul HCCA *Scheduler*

Komponen utama dari arsitektur *simulator* HCCA adalah modul HCCA *scheduler*. Tidak seperti operasi pada *network layer*, *scheduler* beroperasi pada MAC *layer* sangat tergantung pada *layer* di bawahnya. Jadi, kita harus mendefinisikan *interface* umum yang cukup untuk mengadopsi urutan algoritma *scheduling*.

Bila dalam keadaan aktif, baik penjadwalan QSTA dan QAP berpindah-pindah antara dua *state* utama:

- *IDLE*. Merupakan *passive state*: peket-paket diantrikan dari *downlink Traffic Stream* (TS) tidak dikirimkan menggunakan HCCA dan bukan *poll* yang dibangkitkan (hanya QAP). Hanya dua aksi yang diikuti oleh keadaan ini: (i) mengantrikan paket *downlink* baru dari TS yang dibentuk, (ii) menambahkan TS baru ke kumpulan TS yang dibentuk.
- *BUSY*. Merupakan *active state*. Semua tindakan yang diijinkan pada *state* sebelumnya dapat berlangsung pada *state* ini. Selain itu, paket-paket *downlink* dari TS yang dibentuk dapat dikirimkan atau QAP dapat mem-*poll* QSTA.

Sedangkan fungsi-fungsi berikut ditentukan untuk algoritma penjadwalan yang digunakan pada QAP dan QSTA:

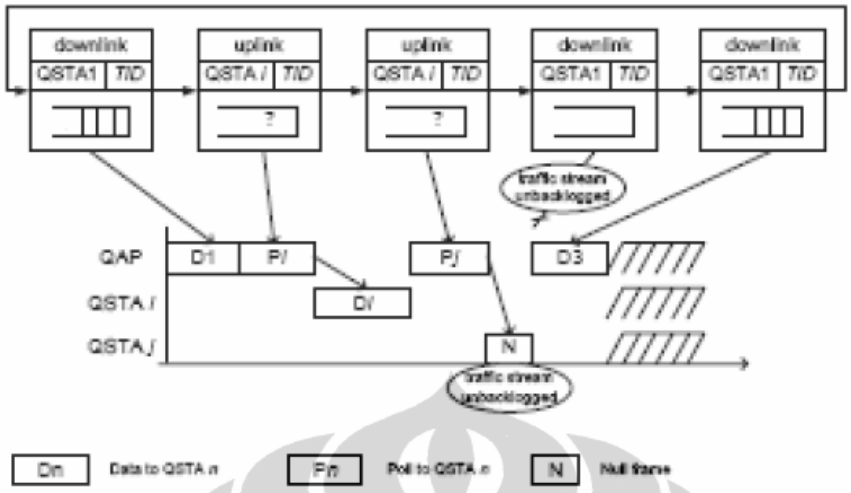
- *Enque()*. Fungsi ini dipanggil oleh *link layer* dimana paket *downlink* baru telah diterima dari *layer* di atasnya. Paket yang datang harus diantrikan pada struktur data *scheduler* tertentu dari modul HCCA *scheduler*. Ada

kasus dimana *scheduler* dapat membuang paket, sebagai contoh bila *buffer* dialokasikan ke paket *downlink* terbatas.

- *Deque*(). Fungsi ini dipanggil oleh MAC bila stasiun memiliki kontrol terhadap medium. Pengaruhnya tergantung kepada algoritma *scheduling* dan status saat ini dari *scheduler*. Tindakan yang diijinkan termasuk: (i) *scheduler* melewatkan ke MAC paket *downlink* yang dikirimkan, (ii) *scheduler* berkomunikasi dengan MAC bahwa QSTA yang diberikan harus di-*poll* dengan durasi TXOP tertentu, (iii) sebuah *poll* terhadap dirinya sendiri harus dikirim, (iv) tidak ada tindakan yang dilakukan. Sudah tentu, tindakan (ii) dan (iii) hanya dilakukan oleh *scheduler* QAP.
- *Get_next_cap*(). [hanya QAP] Fungsi ini dipanggil oleh MAC bila *scheduler* QAP kehilangan kontrol terhadap medium. Digunakan oleh *scheduler* untuk memulai awal MAC dan CAP berikutnya. Sampai waktu itu, fungsi HCCA dari QAP adalah *idle*.
- *addTSPEC*(). [hanya QAP] Fungsi ini dipanggil oleh MAC bila pemberian ijin deretan trafik baru diminta.
- *Get_queue_size*(). [hanya QSTA] Fungsi ini dipanggil oleh MAC sebelum mengirimkan *frame* data. Ia mengembalikan ukuran antrian dari TS tertentu. Informasi ini di-*piggybacked* ke semua *frame* data yang dibangkitkan oleh QSTA yang memiliki TS tertentu.

3.1.1.4 *Reference Scheduler*

Reference scheduler merupakan suatu penjadwalan yang terdapat pada QAP seperti yang digambarkan pada bab 2. Gambar 3.2 menunjukkan struktur data utama yang digunakan oleh *reference*, sebagai contoh, daftar *descriptor*. Masing-masing *descriptor* mengandung informasi yang berhubungan dengan TS yang dibentuk. Di dalam kasus TS *downlink*, *descriptor* juga menyimpan antrian transmisi yang berhubungan.



Gambar 3.2. Contoh *Reference Scheduler*

Penerapan metode *interface* antara modul MAC dan HCCA *scheduler* adalah:

- *Enque*(). Metode ini menambahkan paket ke dalam antrian transmisi yang benar.
- *Deque*(). Bila ini adalah *descriptor* terakhir dalam daftar, maka QAP menahan diri untuk mengakses medium menggunakan HCCA sampai SI berikutnya. Bila ini adalah *descriptor* saat ini bukan yang terakhir, kemudian cek arah dari TS: (i) bila *downlink* dan antrian transmisi tidak kosong, kirim sebanyak mungkin *frame* yang diijinkan oleh durasi maksimum dari TS, (ii) bila *uplink*, *poll* QSTA yang berhubungan untuk durasi yang sama dengan maksimum TXOP yang dijamin ke TS.
- *addTSPEC*(). Metode ini membentuk prosedur *admission control* yang digambarkan pada bab 2. bila deretan trafik diijinkan, ia membuat *descriptor* baru dan menyelipkannya dalam daftar circular.
- *Get_next_cap*(). Metode ini hanya mengembalikan waktu yang tersisa ke SI berikutnya.

3.1.1.5 *Oneflow Scheduler*

Oneflow scheduler merupakan suatu mekanisme penjadwalan yang digunakan QSTA yang hanya dapat membentuk satu *uplink* TS dengan QAP.

Paket-paket *outgoing* diatur dalam *First-In-First-Out*. Dengan asumsi ini, metode-metode yang dibutuhkan dapat diterapkan sebagai berikut:

- *Enque*(). Metode ini menambahkan paket yang diterima dari *Link Layer* ke ujung antrian transmisi.
- *Deque*(). Metode ini mengembalikan paket pada awal antrian transmisi.
- *addTSPEC*(). Metode ini mengecek apakah tidak ada TS yang dibentuk sebelumnya.
- *Get_queue_size*(). Metode ini mengembalikan jumlah ukuran paket-paket yang diantrikan.

3.1.2 *Routing* pada NS-2

Pada penelitian ini, penulis memfokuskan pada skenario dengan topologi dengan topologi jaringan *Basic Service Set* (BSS). Beberapa stasiun berkomunikasi secara langsung dengan *Access Point* (AP) melalui *link wireless*.

Tentu saja, *wireless routing* diperlukan di sini. Sayangnya, dalam NS-2 standar *wireless routing*-nya adalah *ad-hoc*. Ada beberapa skema *routing* yang telah diterapkan pada NS-2, diantaranya*: Destination-Sequenced Distance-Vector* (DSDV), *Ad-hoc On-demand Vector* (AODV) dan *Dynamic Source Routing* (DSR). Namun, *ad-hoc routing* memungkinkan transmisi *multi-hop*, yang tidak realistik untuk topologi BS atau infrastruktur WLAN. Oleh karena itu, tidak satupun dari skema *routing* di atas sesuai untuk penelitian ini yang menggunakan topologi BSS. Di sini, NOAH (*No Ad-hoc Routing Agent*) yang menggunakan *static routing* digunakan. NOAH adalah ekstensi *routing* NS-2, yang memberikan *routing* hirarki secara langsung. *Routing* ini hanya memungkinkan komunikasi langsung antara stasiun-stasiun (*mobile nodes*) dan AP, dan tidak memerlukan paket-paket *routing* yang harus dikirim.

3.2 Model Simulasi

Untuk membandingkan kinerja IEEE 802.11e HCCA pada *Wireless Local Area Network*, serangkaian simulasi dilakukan. Pada simulasi ini, beberapa stasiun disusun sedemikian rupa sehingga berada dalam batasan agar dapat berkomunikasi dengan AP. Dalam melakukan perbandingan kinerja protokol

MAC HCCA ini, penulis membandingkannya dengan protokol MAC standar *Distributed Coordination Fuction* (DCF). Hal ini dilakukan karena protokol MAC DCF merupakan protokol MAC standar yang saat ini ada pada WLAN.

3.2.1 Parameter Simulasi

Untuk mendapatkan hasil simulasi yang tepat terhadap sistem yang akan dilakukan, maka perlu ditentukan beberapa parameter simulasi. Simulasi MAC protokol IEEE 802.11e pada WLAN menggunakan *physical layer* IEEE 802.11b yang memakai modulasi *Direct Sequence Spread Spectrum* (DSSS) dengan kecepatan transmisi datanya mencapai 11 Mbps. Sedangkan semua parameter-parameter MAC ditentukan dari *physical layer* tersebut [7]. Ukuran *Congestion Window* minimumnya adalah 31 dan *Congestion Window* maksimumnya adalah 1023. Tabel 3.1 menunjukkan beberapa parameter MAC untuk *physical layer* IEEE 802.11b.

Tabel 3.1. Parameter MAC untuk *physical layer* IEEE 802.11b

Karakteristik	Nilai
Slot Time	20 µs
SIFS Time	10 µs
Data Rate	11 Mbps
Basic Rate	1 Mbps
Preamble Length	144 bit
PLCP *Header Length*	48 bit
PLCP *Data Rate*	1 Mbps
CWmin	31
CWmax	1023

Semua simulasi dijalankan selama 100 detik waktu simulasi namun tidak ada data yang dikirimkan untuk 20 detik pertama. Waktu simulasi ini sama seperti yang dilakukan oleh I. Inan, F. Keceli dan E. Ayanoglu [8].

Model propagasi yang dilakukan pada simulasi ini adalah model propagasi *two-ray ground*. Model propagasi ini tidak hanya mempertimbangkan lintasan langsung antara dua *node* namun juga mempertimbangkan lintasan *ground reflection*. Antena yang digunakan adalah antena omni. Simulasi dilakukan pada koordinat sehingga AP berada tepat di tengah-tengah stasiun yang lain. Untuk itu, AP diletakkan di pusat lingkaran yang memiliki radius 10. Panjang area yang digunakan adalah 100. Jenis *interface* antrian yang digunakan adalah *DropTail/PriQueue* dengan jumlah paket dalam antrian dibatasi 50 paket untuk setiap node atau stasiun di dalam sistem. Jumlah ini dipandang cukup memadai karena beberapa peelitian menunjukkan bahwa hanya terjadi perbedaan kinerja yang sedikit antara jumlah antrian paket 50 dan 100.

Untuk parameter MAC-nya, simulasi ini menggunakan *Congestion Window* minimum (CWmin) 31 dan *Congestion Window* maksimum (CWmax) 1023. Ukuran *Congestion Window* sangat tergantung pada *physical layer* yang digunakan.

3.2.2 Model Trafik

Simulasi yang dikonfigurasikan pada masing-masing stasiun menggunakan sumber trafik *bidirectional*. Ada dua jenis trafik yang diberikan pada masing-masing skenario yaitu trafik audio, video dan data.

Trafik audio menggunakan *generator* trafik *Constant Bit Rate* (CBR) yang membangkitkan data secara kontinyu dengan *bit rate* konstan. Trafik audio CBR ini dipasang pada *agent transport Universal Datagram Protocol* (UDP). Dengan cara yang sama diterapkan oleh D. Chen, D. Gru, J. Zhang [6], trafik audio ini diparameterkan ke model *voice* G.711, dimana paket *voice* dibangkitkan dengan laju bit yang konstan 64 kbps, ukuran paket 160 bytes dan waktu antar kedatangan paket 20 ms. Skema G.711 dipilih karena masih umum digunakan, karena kesederhanaannya disamping skema dengan kompresi yang lebih baik.

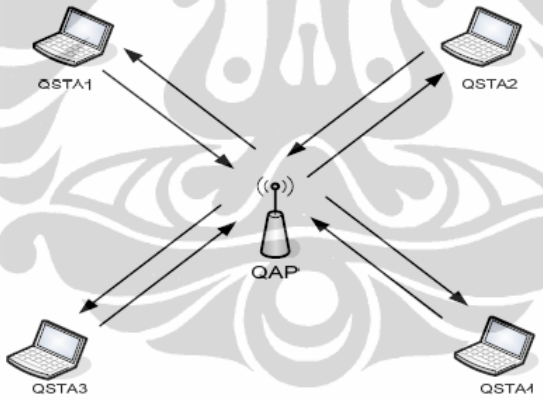
Trafik video juga menggunakan *generator* trafik CBR dengan *agent transport* yang sama dengan trafik audio yaitu UDP. Model trafik video ini dibangkitkan dengan laju bit yang konstan 80 kbps, dengan ukuran paket sebesar 1280 bytes dan dengan waktu *interval* paket 16 ms.

Dalam sebagian skenario, penulis juga menggunakan sumber trafik *best effort* pada simulasi trafik audio dan video. Tidak seperti trafik audio dan video yang menggunakan *generator* trafik CBR, trafik BE diterapkan pada *generator* trafik eksponensial seperti yang dilakukan oleh D. Chen, D. Gru dan J. Zhang [6]. Ukuran paket yang digunakan adalah rata-rata 210 bytes dengan laju pengiriman 64 kbps. Sedangkan waktu kedatangannya adalah 25 ms [6]. Trafik BE ini diberikan pada *agent* TCP.

3.2.3 Skenario Simulasi

3.2.3.1 Skenario 1

Skenario pertama ini digunakan untuk melihat dukungan protokol MAC HCCA pada aplikasi trafik audio dan video. Topologi yang digunakan untuk skenario pertama ini adalah seperti yang diperlihatkan pada Gambar 3.3 berikut ini.



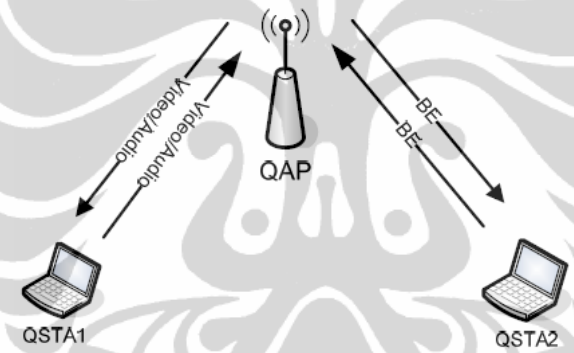
Gambar 3.3. Topologi Skenario 1

Dari topologi tersebut dapat dilihat bahwa ada empat stasiun yang masing-masing mengirimkan trafik audio ke AP. Trafik yang dikirimkan bisa secara *uplink* maupun *downlink* untuk memperlihatkan mekanisme seperti kondisi nyatanya. Selanjutnya, dengan topologi dan jumlah stasiun yang sama, trafik yang dikirimkan diganti dengan trafik video. Seperti trafik audio, trafik video ini juga dikirimkan secara *bidirectional*.

Untuk skenario yang sama juga akan dilakukan simulasi pada protokol MAC 802.11 mode DCF sebagai perbandingan terhadap kinerja protokol 802.11e HCCA.

3.2.3.2 Skenario 2

Berbeda dengan skenario 1, skenario 2 menghadirkan trafik *Best Effort*. Hal ini dilakukan untuk memperlihatkan pengaruh yang terjadi apabila pada suatu jaringan WLAN selain kita menggunakan trafik audio atau video, kita juga menggunakan trafik BE ini. Topologi dari skenario 2 ini dapat dilihat pada Gambar 3.4. Jumlah stasiun yang mengirimkan trafik BE divariasikan untuk memperlihatkan pengaruh yang terjadi.

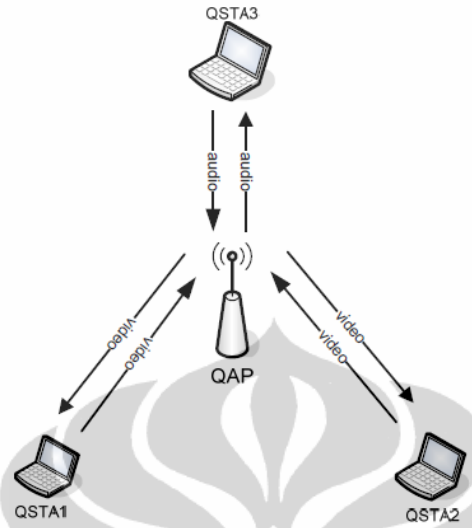


Gambar 3.4. Topologi Skenario 2

Untuk skenario ke-2 ini juga akan dilakukan simulasi menggunakan protokol MAC 802.11 mode DCF.

3.2.3.3 Skenario 3

Skenario 3 menggabungkan trafik audio dan video pada suatu jaringan WLAN. Pada skenario ini, ada dua stasiun mengirimkan trafik video pada arah *uplink* dan *downlink*. Sedangkan trafik audio dibangkitkan oleh satu stasiun yang *bidirectional*. Skenario 3 ini dilakukan untuk memperlihatkan mekanisme protokol MAC HCCA dalam menghadapi dua sumber trafik multimedia. Sebagai pembanding, juga akan dilakukan simulasi menggunakan protokol MAC 802.11 mode DCF. Gambar 3.5 menunjukkan topologi skenario 3.



Gambar 3.5. Topologi Skenario 3

3.2.4 Metrik Kinerja

Di dalam simulasi ini, ada beberapa ukuran kinerja yang diambil sebagai acuan dalam membandingkan protokol MAC HCCA dengan protokol MAC standar DCF. Ukuran atau metrik kinerja yang pertama adalah *jitter*. *Jitter* merupakan nilai waktu dari satu pake ke paket berikitnya dalam sebuah trafik.

Ukuran kinerja lainnya adalah *throughput*. *Throughput* merupakan jumlah paket yang sukses diterima dalam satu satuan waktu. Satuan yang dipakai di sini adalah bps (bit per second).