QualNet
Building Smarter Networks

QualNet Hacks

＃３ 物理層と伝搬チャネル

改訂1版

About QualNet Hacks

QualNet は、ほぼ全てソースコードが公開されています。

これらのソースコードには QualNet の内部を理解するために有益な情報が沢山ちりばめられています。

しかしながら、ソースコードの量は莫大であり、その内容を簡単に理解することが難しいのも事実です。

本書は、QualNet 内部の理解をより深めて頂くことを目的として作成しました。

本書を手掛かりにして、より一層 QualNet 活用して頂ければ幸いです。

このドキュメントは QualNet5.1 のソースコードに準拠します

【ソースコードに関する注意事項】
本ドキュメントには、ソースコードの一部が複写されています。ソースコードの使用に関しては、以下の開発元の制限に則りますので、ご注意下さい。

改訂履歴

版	日付	内容
改訂 1 版	2012/1/25	TBD となっていた以下の 3 節の本文を追記。 3.3.3.1 受信信号電力(S)の算出 3.3.3.2 受信干渉電力(I)の算出 3.3.3.3 雑音電力(N)の算出

contents

3 物理層と伝搬チャネル

本章では、物理層における送受信処理ならびに無線伝搬チャネルを QualNet がどのようにモデル化しているかについて解説する。まず 3.1 節で QualNet のチャネルモデルについて解説し、続いて 3.2 節では QualNet の無線リンクではどのようなリンクバジェット要素が考慮されているかについて言及する。その後、3.3 節では物理層における送信処理と受信処理のモデル化について、3.4 節ではアンテナモデルについて解説し、最後に 3.5 節で無線伝搬路のモデル化方法を述べる。

3.1 チャネルモデル

QualNet のチャネルモデルには、現実世界における周波数チャネルとは大きく異なる点が 1 点存在する。それは、QualNet のチャネルモデルには連続した周波数領域という概念がないという点であり、そのため、チャネル間に干渉が起こり得ない作りとなっている。QualNet を使用する際にはまずこの点を十分に理解しておく必要があるので、この点について以下でもう少し詳しく説明する。

3.1.1 現実世界における周波数チャネル

現実世界においては、図 3-1 に示すように、ある周波数を中心にある帯域幅を持った周波数領域のことを一般にチャネルと呼んでいる。個々のチャネルは周波数帯が被る場合もあれば被らない場合もあるが、これは個々の無線システムにおいてチャネル配置をどのように規定しているかに依存する。その意味では、チャネルとは周波数領域におけるある特定区間を表す用語であると言える。そのため、定義領域（定義区間）が被るチャネルは当然ながら互いに干渉し合う。

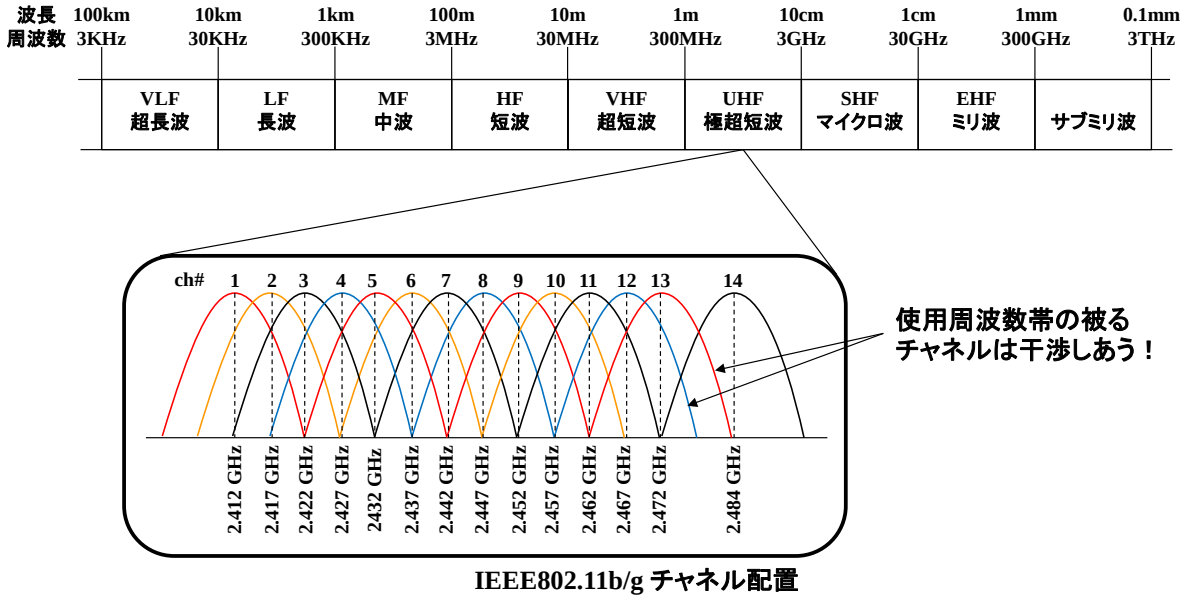


図 3-1 現実世界の周波数チャネル

3.1.2 QualNet のチャネルモデル

一方で QualNet の世界では、連続した周波数領域という概念は存在しておらず、図 3-2 のようにチャネルという名の互いに独立なパイプを用意して各パイプに中心周波数値や伝搬路特性などの属性を与えるようなモデルとなっている。（ここで「パイプ」という用語はあくまで比喩

的表現として用いている。）QualNet には連続した周波数領域という概念が存在しないため、たとえ同一の中心周波数値が設定された複数のチャネルが存在する場合でも、それらのチャネルは互いに干渉することはない。QualNet の各チャネルに設定される中心周波数値は、連続した周波数領域における 1 点を表すのではなく、あくまで伝搬減衰を計算する際のパラメータに過ぎないという点を注意しておきたい。

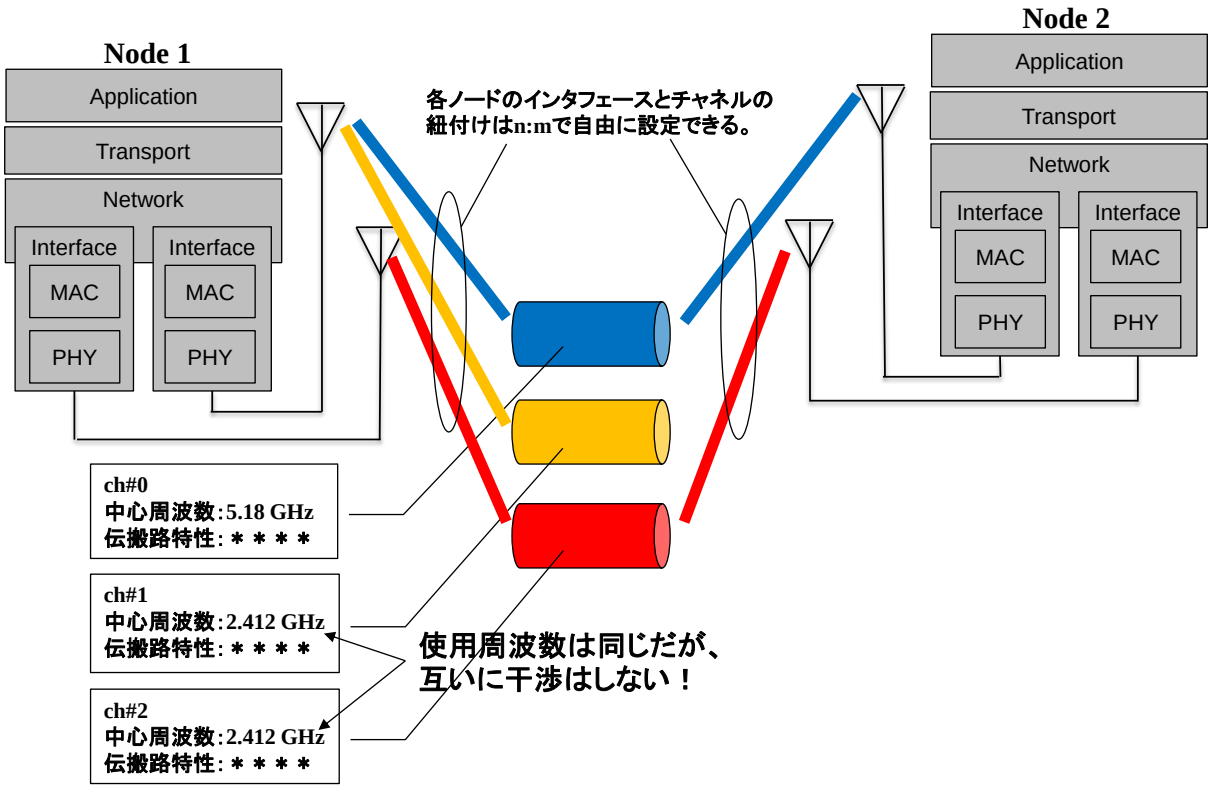


図 3-2　QualNet のチャネルモデルイメージ

3.1.3 チャネルモデルの実装

図 3-3 は、QualNet のチャネルモデル周辺の実装イメージである。QualNet では、定義されたチャネルの集合は、PropChannel インスタンスの配列として取り扱われる。配列の各要素は各チャネルを表しており、各 Node インスタンスからは node->propChannel ポインタを辿ってこの配列、つまり各チャネルの情報にアクセスすることができるようになっている。以下の設定例のような各チャネルに対する設定が、各 PropChannel インスタンスに反映される。設定におけるカギ括弧 [ ] 内の数字はチャネル番号であり、PropChannel 配列における配列インデックスに相当している。

```
PROPAGATION-CHANNEL-FREQUENCY[0] 2400000000 ※チャネル 0 の場合は”[0]”の記述は省略可
PROPAGATION-PATHLOSS-MODEL[0] FREE-SPACE
PROPAGATION-SHADOWING-MODEL[0] CONSTANT
PROPAGATION-FADING-MODEL[0] RAYLEIGH

PROPAGATION-CHANNEL-FREQUENCY[1] 5100000000
PROPAGATION-PATHLOSS-MODEL[1] OKUMURA-HATA
PROPAGATION-SHADOWING-MODEL[1] NONE
PROPAGATION-FADING-MODEL[1] NONE

PROPAGATION-CHANNEL-FREQUENCY[2] 500000000
PROPAGATION-PATHLOSS-MODEL[2] TWO-RAY
PROPAGATION-SHADOWING-MODEL[2] LOGNORMAL
PROPAGATION-FADING-MODEL[2] RICEAN
```

また Node インスタンスには、このチャネルインスタンスへの参照ポインタ以外に、伝搬データインスタンスへのポインタ node->propData も保持しており、これを辿ることで各チャネルからの受信電波の情報（PropData インスタンス）にアクセスすることができるようになっている。

PropChannel インスタンス同様に、この PropData インスタンスもチャネル数分の配列として全体が構成されており、これらの配列インデックスはその並び順に関して整合がとれている。つまり、node->propChannel[1]と node->propData[1]は同一チャネルに関する情報を指しており、node->propChannel[1]はチャネルそのものの情報を、node->propData[1]はそのチャネル上で受信した電波の情報を表している。）ただし、PropChannel 配列はシミュレーション全体で 1 つしか存在しないが、PropData 配列はノード毎に実体が定義されるので注意したい。また、PropData インスタンスの phyListening 変数は、配列サイズが MAX_NUM_PHY（定義値は 28）の BOOL 型変数配列を参照している。この配列により、当該チャネルを Listen している PHY インスタンスがどれであるかが一目で分かるようになっている。

図中には、これら以外に PHY インスタンスも登場する。PHY インスタンスについては、Node インスタンスからは node->phyData ポインタを辿ってアクセスできる。phyData ポインタは正確には PHY インスタンスへのポインタの配列（PhyData*配列）へのポインタである。node->phyData が指すこの PhyData*配列は、MAX_NUM_PHY（定義値は 28）個の PhyData ポインタの配列であり、ノード毎にその実体が定義される。先頭から順番に実際に各ノードで定義された PHY インスタンス（PhyData インスタンス）のアドレスがセットされ、PHY インスタンス未定義分については NULL 値がセットされるようになっている。また、PhyData インスタンスの channelListenable 変数と channelListening 変数は、それぞれ配列サイズが定義済みチャネル数である BOOL 型変数配列を参照している。この配列により、当該 PHY インスタンスが Listen しているチャネルがどれであるかが一目で分かるようになっている。

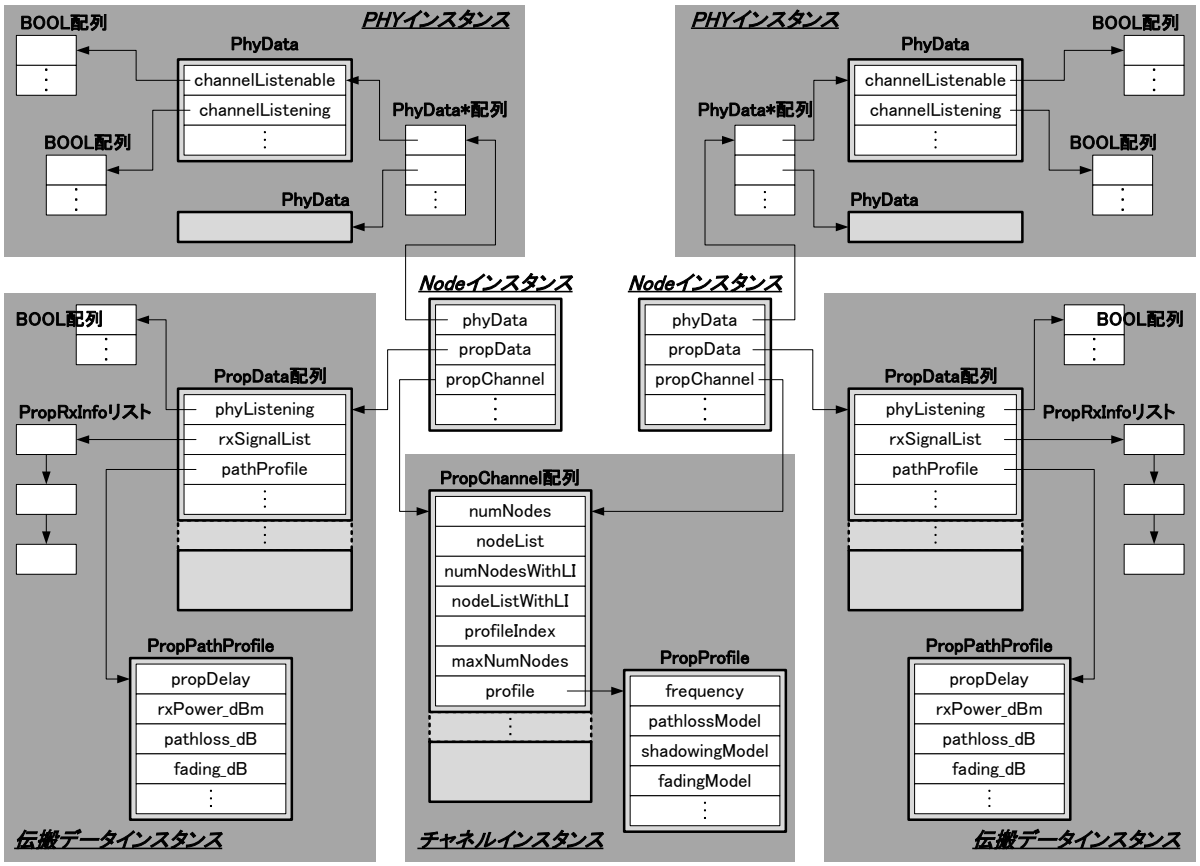


図 3-3　QualNet のチャネルモデル実装

なお、図 3-3 に登場する各構造体の内容を、それぞれ表 3-1 から表 3-5 に示す。

表 3-1 PropChannel 構造体

メンバ変数名	データ型	説明
numNodes	int	当該チャネルを介した送受信が可能なノードの数
nodeList	Node **	当該チャネルを介した送受信が可能なノードのリスト
numNodesWithLI	int	Limited Interference グループに設定されているノードの数
nodeListWithLI	Node **	Limited Interference グループに設定されているノードのリスト
profileIndex	int	チャネルプロファイルのインデックス値
maxNumNodes	int	未使用
profile	PropProfile *	チャネルプロファイル情報

表 3-2 PropProfile 構造体

メンバ変数名	データ型	説明
profileIndex	int	チャネルプロファイルのインデックス値
propLimit_dB	double	伝搬可能な受信電力の上限値
propMaxDistance	D_Float64	伝搬可能な送受信機間距離の上限値
propCommunicationProximity	D_Float64	チャネルプロファイルの更新解像度（この距離以上移動するとチャネルプロファイルが更新される）
propProfileUpdateRatio	D_Float64	チャネルプロファイル更新比率
frequency	double	当該チャネルの中心周波数
antennaHeight	double	無線機設置面からのアンテナ高
wavelength	double	波長
pathlossModel	PathlossModel	パスロスモデル種別
elevationSamplingDistance	float	標高データのサンプリング間隔(距離)
climate	int	気候種別
refractivity	double	屈折率
conductivity	double	伝導率
permittivity	double	誘電率
humidity	double	湿度
polarization	int	偏波面の向き（Horizontal(0) / Vertical(1)）
polarizationString[5]	char 配列	偏波面の向き（文字列表記）
shadowingModel	ShadowingModel	シャドウイングモデル種別
shadowingMean_dB	double	シャドウイングによる平均減衰量
fadingModel	FadingModel	フェージングモデル種別
kFactor	double	フェージングの K ファクター
dopplerFrequency	double	ドップラー周波数
baseDopplerFrequency	double	Gaussian Component データのサンプリング時のドップラー周波数
samplingRate	D_Int32	Gaussian Component データのサンプリングレート
numGaussianComponents	int	Gaussian Component の数
gaussianComponent1	double *	Gaussian Component の実数成分値
gaussianComponent2	double *	Gaussian Component の虚数成分値
numChannelsInMatrix	int	パスロスマトリクスデータ内のチャネル数
channelIndexArray	int *	パスロスマトリクスデータ内のインデックス配列
numNodesInMatrix	int	パスロスマトリクスデータ内のノード数
matrixList	vector <pathLossMatrixValue>	パスロスマトリクスのデータ
propGlobalVar	void *	特に使用されている形跡は見当たらない
monthofyear	double	ASAPS モデルで使用
dayofmonth	double	ASAPS モデルで使用
timeofday	double	ASAPS モデルで使用
Tindex	double	ASAPS モデルで使用
mintoa	double	ASAPS モデルで使用
hfTxpower	double	ASAPS モデルで使用
reqSnr	double	ASAPS モデルで使用
manMadenoise	double	ASAPS モデルで使用
hfbandwidth	double	ASAPS モデルで使用

reqPercentageDay	double	ASAPS モデルで使用
TxantennaData	void *	ASAPS モデルで使用
RxantennaData	void *	ASAPS モデルで使用
asapsPath[ ]	char 配列	ASAPS モデルで使用
motionEffectsEnabled	BOOL	高速レイリーフェージングモデルの場合に true
propagationEnvironment	PropagationEnvironment	OKUMURA_HATA モデル, COST231_Hata モデル, COST231_Walfish_Ikegami モデル, ITU-R モデルで使用
roofHeight	double	COST231_Walfish_Ikegami モデルで使用
streetWidth	double	COST231_Walfish_Ikegami モデルで使用
buildingSeparation	double	COST231_Walfish_Ikegami モデルで使用
RelativeNodeOrientation	double	URBAN-AUTOSELECT-MODEL で使用
MaxRoofHeight	double	URBAN-AUTOSELECT-MODEL で使用
MinRoofHeight	double	URBAN-AUTOSELECT-MODEL で使用
losIndicator	LoSIndicator	Street_Microcell model, Indoor で使用
Num_builings_in_path	int	Street_M_to_M model で使用
suburbanTerrainType	SuburbanTerrainType	Suburban-Foliage Model で使用
pathlossModelPrimary	PathlossModel	PL_OPAR モデルで使用
obstructions	Obstruction *	PL_OPAR_PROP で使用
numObstructions	int	PL_OPAR_PROP で使用
pathlossArea	PathlossArea *	未使用
numPathlossAreas	int	未使用
constructionMaterials	TERRAIN::Construction Materials	ITU-R Indoor モデルで使用

表 3-3 PropData 構造体

メンバ変数名	データ型	説明
numPhysListenable	int	当該チャネルに Listenable マスクを設定している PHY インスタンスの数
numPhysListening	int	当該チャネルに Listening マスクを設定している PHY インスタンスの数
phyListening	BOOL *	各 PHY インスタンスが当該チャネルに Listening マスクを設定しているかどうかを示すフラグの配列
limitedInterference	BOOL	Limited Interference グループが有効かどうか
shadowingDistribution	RandomDistribution<double>	シャドウイングによる減衰値の乱数列
nodeListId	int	不明。ノード数を足しこんでいる。
numSignals	int	受信シグナル数
rxSignalList	PropRxInfo *	受信シグナルのリスト
fadingStretchingFactor	double	フェージングのストレッチング係数
pathProfile	PropPathProfile *	伝搬パスプロファイル情報
propVar	void *	特に使用されている形跡は見当たらない
numPathLossCalculation	int	未使用

表 3-4 PropPathProfile 構造体

メンバ変数名	データ型	説明
propDelay	clocktype	伝搬遅延
distance	double	送受信機間距離
txDOA	Orientation	不明
rxDOA	Orientation	不明
rxPower_dBm	double	受信電力 [dBm]
pathloss_dB	double	パスロスによる減衰量 [dB]
fading_dB	double	フェージングによる減衰量 [dB]
channelReal	double	未使用
channelImag	double	未使用
rxFrequency	double	受信機側の周波数
fromPosition	Coordinates	送信機の座標

toPosition	Coordinates	受信機の座標
sequenceNum	int	伝搬パスプロファイル情報を識別するためのシーケンス番号
weatherPathloss_dB	double	気象変動による減衰量 [dB]
weatherSequenceNum	int	不明

表 3-5 PropRxInfo 構造体

メンバ変数名	データ型	説明
txNodeId	NodeAddress	送信側ノード ID
txPhyIndex	short	送信側ノードにおける送信 PHY インデックス
rxStartTime	clocktype	受信開始時刻
duration	clocktype	受信期間
distorted	BOOL	特に使用されている形跡は見当たらない
rxPower_dBm	double	受信電力 [dBm]
pathloss_dB	double	パスロスによる減衰量 [dB]
fading_dB	double	フェージングによる減衰量 [dB]
channelIndex	int	チャネルインデックス
channelReal	double	未使用
channelImag	double	未使用
frequency	double	中心周波数
rxDOA	Orientation	不明
txMsg	Message *	送信メッセージ（当該電波で送られているはずのデータ）
prev	PropRxInfo *	受信シグナルリスト管理用フィールド
next	PropRxInfo *	受信シグナルリスト管理用フィールド

3.1.4 使用チャネルの設定

QualNet シナリオファイルでは、*PHY-LISTENABLE-CHANNEL-MASK* 及び *PHY-LISTENING-CHANNEL-MASK* でマスクをかけることで、各チャネルに対する受信可否を設定できる。それぞれの設定項目の意味は以下の通りである。

表 3-6 使用チャネル設定

設定項目	説明
PHY-LISTENABLE-CHANNEL-MASK	受信可能なチャネルを設定
PHY-LISTENING-CHANNEL-MASK	現在実際に受信しているチャネルを設定 （Simulation 実行中にプログラム内で変更可能）

設定は、最上位 bit が最初のチャネルで最下位 bit が最後のチャネルに対する受信マスクを表すビットストリングで表現する。例えば以下は 0 番目と 1 番目のチャネルを受信可能とする設定である。

```
PHY-LISTENABLE-CHANNEL-MASK 110
```

なお、これらの設定はインタフェース単位にも設定でき、この場合は以下の例のように、どのインタフェースに対する設定であるかを IP アドレスで指定する。

```
[1] IP-ADDRESS[0]  192.168.0.1
[1] IP-SUBNET-MASK[0]  255.255.0.0
[1] IP-ADDRESS[1]  10.31.203.1
[1] IP-SUBNET-MASK[1]  255.255.255.0
[ 192.168.0.1 ] PHY-LISTENABLE-CHANNEL-MASK 110
[ 10.31.203.1 ] PHY-LISTENABLE-CHANNEL-MASK 001
```

Listenable と Listening, Transmission チャネルについて

上述のパラメータで設定できる Listenable チャネルと Listening チャネルについてもう少し詳しく説明する。Listening チャネルは、実際に送信された信号を聞くことができるチャネルであり、1 インタフェースに対し複数同時に設定することもできる。現在 Listening チャネルとして設定されている全てのチャネルで送出された信号に対し、受信イベントが発生する(ただし、受信 Node において PROPAGATION-LIMIT 値を超えている必要がある)。Listening チャネルとして設定することができるチャネルが Listenable チャネルである。これらは以下の関数を用いて設定状態の取得・変更が可能である。これらの関数で、前述の PhyData インスタンスの channelListening 変数と PropData インスタンスの phyListening 変数が指し示す先の BOOL 配列の操作が行われる。

```
// 指定インタフェースで指定チャネルを listening できるか取得
BOOL PHY_CanListenToChannel(Node *node, int phyIndex, int channelIndex);
// 指定インタフェースで指定チャネルを listening しているか取得
BOOL PHY_IsListeningToChannel(Node *node, int phyIndex, int channelIndex);
// 指定インタフェースで指定チャネルを listening 開始
void PHY_StartListeningToChannel(Node *node, int phyIndex, int channelIndex);
// 指定インタフェースで指定チャネルを listening 停止
void PHY_StopListeningToChannel(Node *node, int phyIndex, int channelIndex);
```

一方で Transmission チャネルは信号を送信するチャネルで、こちらは 1 インタフェースに対し同時に 1 つしか設定できない。以下の関数を用いて設定状態の変更が可能である。

```
// 指定インタフェースで transmission しているチャネル取得
void PHY_GetTransmissionChannel(Node *node, int phyIndex, int *channelIndex);
// 指定インタフェースで指定チャネルで transmission するよう設定
void PHY_SetTransmissionChannel(Node *node, int phyIndex, int channelIndex);
```

本節のまとめとして、チャネル設定例とそれが反映された状態のイメージを図 3-4 に示す。

```
PROPAGATION-CHANNEL-FREQUENCY[0] 2400000000 ※チャネル 0 の場合は"[0]"の記述は省略可
PROPAGATION-PATHLOSS-MODEL[0] FREE-SPACE
PROPAGATION-SHADOWING-MODEL[0] CONSTANT
PROPAGATION-FADING-MODEL[0] NONE

PROPAGATION-CHANNEL-FREQUENCY[1] 5100000000
PROPAGATION-PATHLOSS-MODEL[1] OKUMURA-HATA
PROPAGATION-SHADOWING-MODEL[1] NONE
PROPAGATION-FADING-MODEL[1] RAYLEIGH

PROPAGATION-CHANNEL-FREQUENCY[2] 500000000
PROPAGATION-PATHLOSS-MODEL[2] TWO-RAY
PROPAGATION-SHADOWING-MODEL[2] LOGNORMAL
PROPAGATION-FADING-MODEL[2] RICEAN

[1] IP-ADDRESS[0]  192.0.0.1
[1] IP-SUBNET-MASK[0]  255.255.255.0
[1] IP-ADDRESS[1]  192.0.1.1
[1] IP-SUBNET-MASK[1]  255.255.255.0
[2] IP-ADDRESS[0]  192.0.0.2
[2] IP-SUBNET-MASK[0]  255.255.255.0
[2] IP-ADDRESS[1]  192.0.1.2
[2] IP-SUBNET-MASK[1]  255.255.255.0

[ 192.0.0.1 ] PHY-LISTENABLE-CHANNEL-MASK 111
[ 192.0.0.1 ] PHY-LISTENING-CHANNEL-MASK 110
[ 192.0.1.1 ] PHY-LISTENABLE-CHANNEL-MASK 111
[ 192.0.0.1 ] PHY-LISTENING-CHANNEL-MASK 111
[ 192.0.0.1 ] PHY-LISTENABLE-CHANNEL-MASK 111
[ 192.0.0.1 ] PHY-LISTENING-CHANNEL-MASK 110
[ 192.0.0.1 ] PHY-LISTENABLE-CHANNEL-MASK 001
[ 192.0.0.1 ] PHY-LISTENING-CHANNEL-MASK 001
```

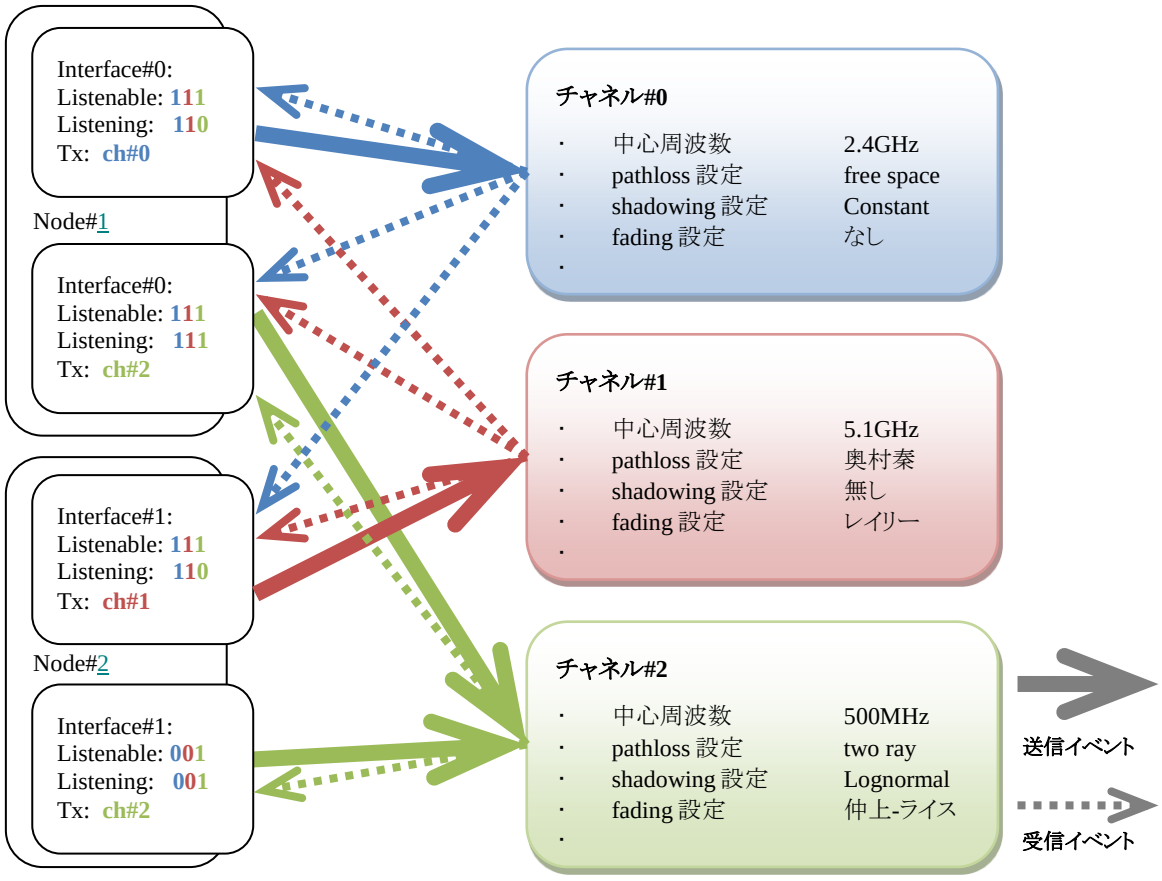


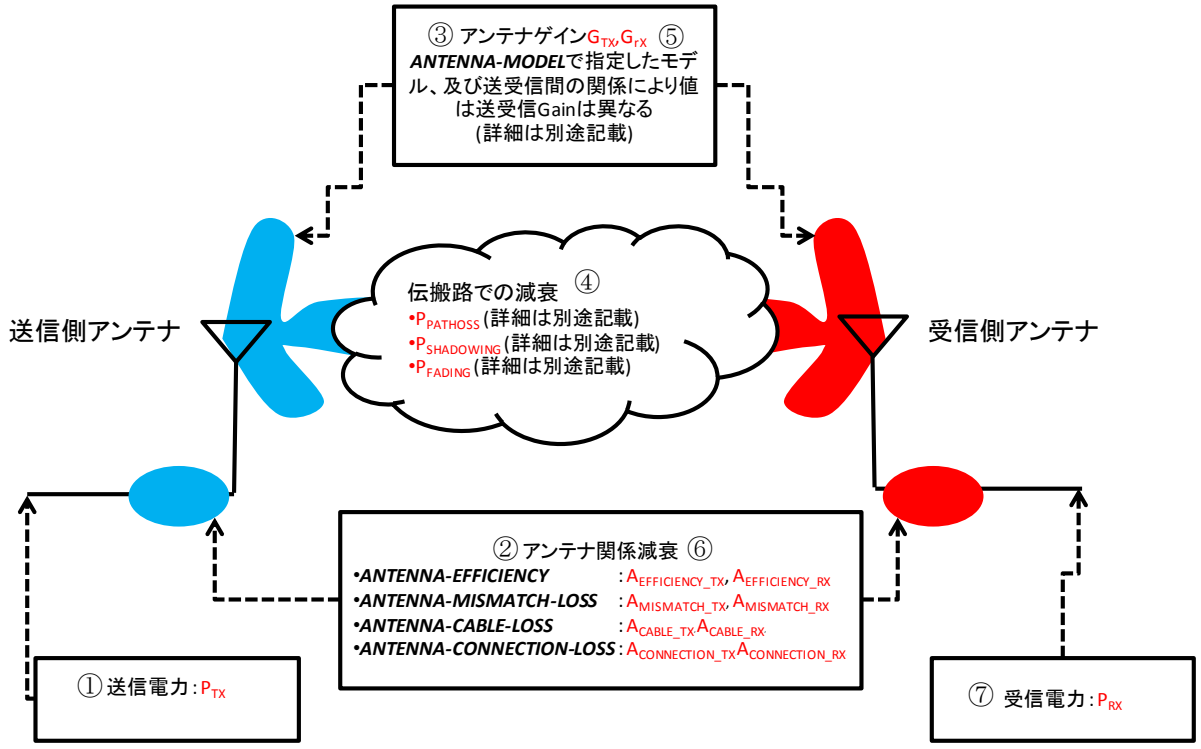
図 3-4 チャネル反映例のイメージ

3.2 リンクバジェット要素

シミュレーションモデル内のある無線リンクに注目した際に、その無線リンクに関して QualNet ではどのようなリンクバジェット要素が設定あるいは計算により考慮されるようになっているのか、という点について本節では解説する。

3.2.1 送受信電力の増幅／減衰に影響する要素

QualNet の無線リンクにおいて考慮されるリンクバジェット要素のうち、送受信電力の増幅／減衰に影響する要素を図 3-5 に示す。



$$
\begin{aligned}
&P_{RX} && : \text{受信電力}\\
&= P_{TX} - (-10\log_{10}(A_{EFFICIENCY\_TX}) + A_{MISMATCH\_TX} + A_{CABLE\_TX} + A_{CONNECTION\_TX}) + G_{TX} && : \text{送信側増幅/減衰}\\
&- P_{PATHOSS} - P_{SHADOWINS} + P_{FADING} && : \text{伝搬路での減衰}\\
&+ G_{RX} - (-10\log_{10}(A_{EFFICIENCY\_RX}) + A_{MISMATCH\_RX} + A_{CABLE\_RX} + A_{CONNECTION\_RX}) && : \text{受信側増幅/減衰}
\end{aligned}
$$

図 3-5　送受信電力の増幅／減衰に影響するリンクバジェット要素

送信側無線機から出力される電波が受信側無線機に達するまでの様子を、以下に時系列で列記する。

① 送信側無線機では、設定あるいは計算されたある大きさの送信電力（P_{TX}）で電波を出力する。

② 送信側無線機内では、まずアンテナ関連のパラメタ設定に応じた減衰を受ける（図の青い楕円部分）。この減衰に関連するパラメタには *ANTENNA-EFFICIENCY* と *ANTENNA-MISMACH-LOSS*、*ANTENNA-CABLE-LOSS*、*ANTENNA-CONNECTION-LOSS* の 4 種類があるが、これらは内部的にはこれらを単純に加算（*ANTENNA-EFFICIENCY* は dB 値に変換したうえでシステムロスから減算）した値をシステムロスとして取り扱っており、減衰計算上は 4 パラメタに分離されている意味は特に無い。（実装上も phyData->systemLoss_dB という 1 変数で管理されている。）

③ アンテナ関連の減衰を受けると、送信電波は、送信側無線機で定義されたアンテナから出力されるが、この時にアンテナゲイン分の増幅を受け実際の出力電力が計算される（図の

青いアンテナ部分）。

④ 出力された電波は伝搬路上の減衰を受ける。伝搬路上で受ける減衰は長区間変動（Pathloss）、短区間変動（Shadowing）、瞬時変動（Fading）の 3 種類に分けて取り扱われる。（図の雲の部分）※実際にはもう 1 種類、気象を考慮した減衰も与えることが可能。

⑤ 伝搬路を通った電波が受信側無線機のアンテナに届くと、受信電力はまずアンテナゲイン分の増幅を受ける。（図の赤いアンテナ部分）

⑥ 続いて受信側無線機におけるアンテナ関連の減衰を受ける（図の赤い楕円部分）。ここでも送信側と同様に 4 つのパラメータの配分比率は何らシミュレーションに影響を与えない。

⑦ 上記②～⑥の減衰／増幅を受けた結果の電力が受信側無線機における受信電力（P_{RX}）となる。

3.2.2 雑音電力ならびに干渉電力の算出のための要素

図 3-5 は送受信電力の増幅／減衰に影響する要素のみを記したが、これに雑音電力と干渉電力を算出するための要素を追記したものを、図 3-6 に示す。

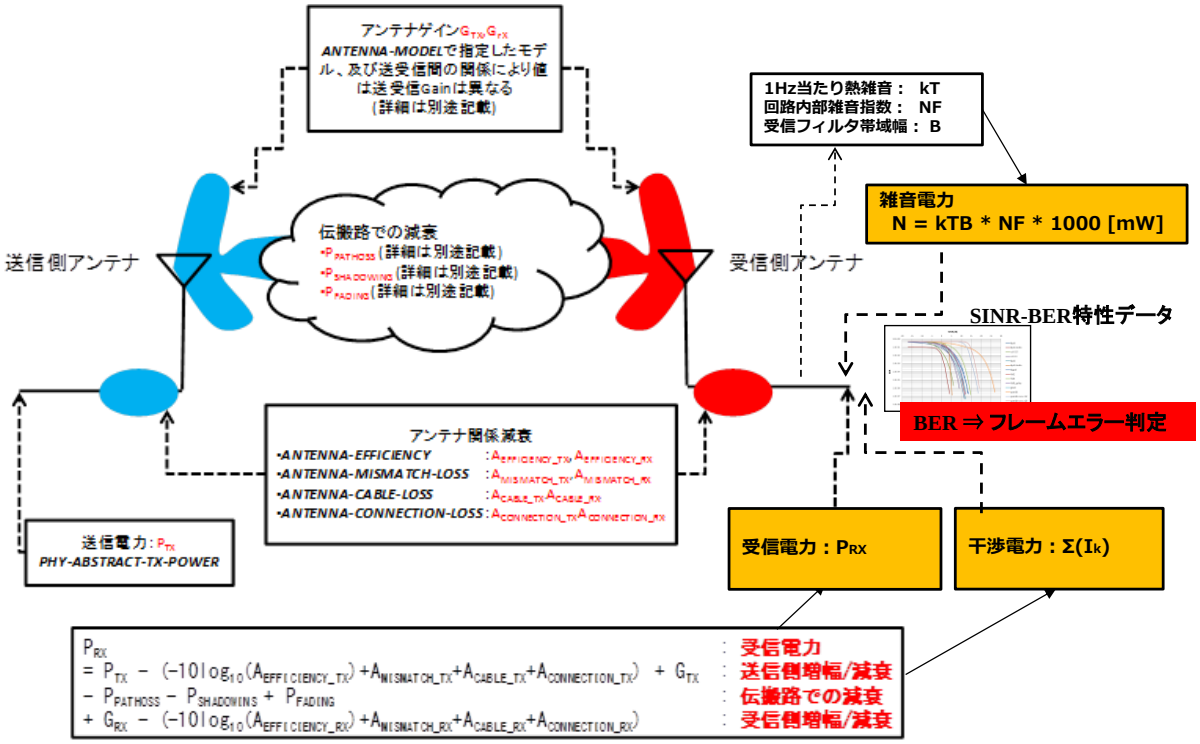


図 3-6 雑音電力および干渉電力算出のためのリンクバジェット要素

雑音は、AWGN として熱雑音および回路内部雑音が考慮されている。熱雑音については 1Hz あたりの雑音電力にベースバンド受信フィルタの通過帯域幅をかけたものとして算出される。内部雑音としてはノイズファクターが設定できるようになっており、トータルの雑音電力は以下の式で計算される値となる。

$$N\ [\mathrm{mW}] = kTB * NF * 1000\ [mW]$$

k：ボルツマン定数（1.38 * 10^-23 [J/K]）
T：温度 [K]
B：ベースバンド受信フィルタの帯域幅 [Hz]
NF：ノイズファクター [dB 値ではなく真値]

干渉電力に関しては、個々の干渉波は希望波の場合と同様の増幅・減衰を受け、それらの足し合わせとしてトータルの干渉電力が算出される。

3.3 PHY 送受信処理モデル

3.3.1 PHY 送信処理

PHY 層における一般的な送信処理の流れを図 3-7 に示す。PHY 層は、MAC 層から MAC フレームを受け取ると（必要であれば PHY ヘッダの付加を行ったうえで）その得られた情報ビット列に対して、誤り訂正符号化処理と拡散処理を実施する。これらの処理を経て生成されたビット列に対してシリアル／パラレル変換を行い、ビット列情報をシンボル情報に変換し、さらに得られたシンボル情報の各成分に帯域制限フィルタをかけてベースバンド信号を生成する。続いてこのベースバンド信号をキャリア変調してキャリア周波数まで引き上げ、RF 部を通してアンテナに出力する。

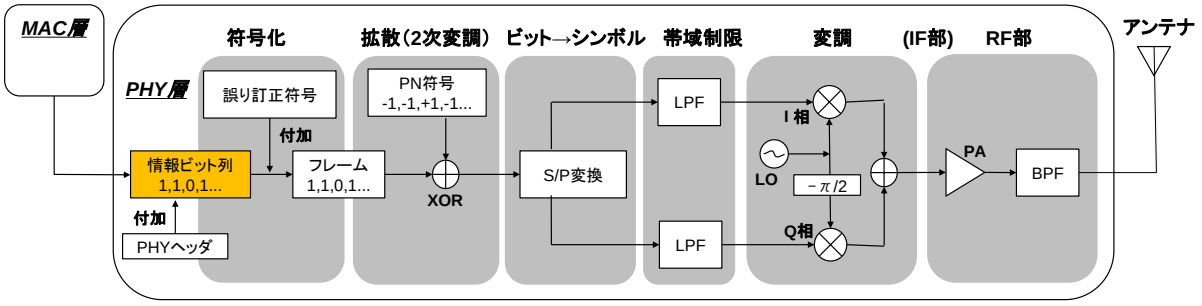


図 3-7 PHY 層における一般的な送信処理の流れ

QualNet の物理層の送信処理では、「PHY ヘッダの付加」以降の情報ビット列に対する一切の処理を省略しており、符号化／拡散／変調方式の影響は受信側で設定する特性データに委ねている。ここで、QualNet では符号化／拡散／変調方式が考慮されていないという意味ではなく、送受信時に実際にプログラム上でビット列に対する操作として符号化／拡散／変調処理は実施していないという意味であるので、誤解を招かないよう注意しておきたい。

実際、送信時には、今送信しようとしているデータ（情報ビット列）に対して、適用する符号化／拡散／変調方式を決定し、決定したデータレートと送信データサイズから信号送信期間を算出し、QualNet カーネルへの信号送信イベントの通知（PROP_ReleaseSignal 関数呼び出し）と信号送信完了イベントの登録（MSG_PHY_TransmissionEnd イベントを引数に指定した MESSAGE_Send 関数呼び出し）を行っている。Abstract PHY モデルの場合の PHY 送信処理の例を以下に示す。

phy_abstract.cpp

```
3839 void PhyAbstractStartTransmittingSignal(
3840     Node* node,
3841     int phyIndex,
3842     Message* packet,
3843     BOOL useMacLayerSpecifiedDelay,
3844     clocktype initDelayUntilAirborne)
3845 {
         ... 中略 ...
3857     clocktype delayUntilAirborne = initDelayUntilAirborne;
3858     PhyData* thisPhy = node->phyData[phyIndex];
3859     PhyDataAbstract* phy_abstract = (PhyDataAbstract*)thisPhy->phyVar;
3860     PhyAbstractStats* stats = &(phy_abstract->stats);
3861     int channelIndex;
3862     Message *endMsg;
3863     int packetsize = 0;
3864     clocktype duration;

         ... 中略 ...
```

```
3957      duration =
3958          PhyAbstractGetFrameDuration(
3959              thisPhy, packetsize, PHY_GetTxDataRate(node, phyIndex));

          ... 中略 ...

3966      PROP_ReleaseSignal(
3967          node,
3968          packet,
3969          phyIndex,
3970          channelIndex,
3971          phy_abstract->txPower_dBm,
3972          duration,
3973          delayUntilAirborne);

          ... 中略 ...

3984
3985      endMsg = MESSAGE_Alloc(node,
3986                             PHY_LAYER,
3987                             0,
3988                             MSG_PHY_TransmissionEnd);
3989
3990      MESSAGE_SetInstanceId(endMsg, (short) phyIndex);
3991      MESSAGE_Send(node, endMsg, delayUntilAirborne + duration + 1);

          ... 中略 ...

4020  }
```

3.3.2 PHY 受信処理

PHY 層における一般的な受信処理の流れを図 3-8 に示す。PHY 層は、アンテナから受信電波の入力を受けると RF 部での処理を経て復調処理を行い、ベースバンド信号を取り出す。さらにベースバンド信号に対して帯域制限フィルタをかけて雑音を除去したうえで、パラレル／シリアル変換を行いシンボル情報をビット列情報に変換（ビット判定）する。そのようにして得られた受信ビット列に対して逆拡散処理と誤り訂正復号化処理を実施して最終的なビット列を得る。（必要であればさらに PHY ヘッダを取り除く。）

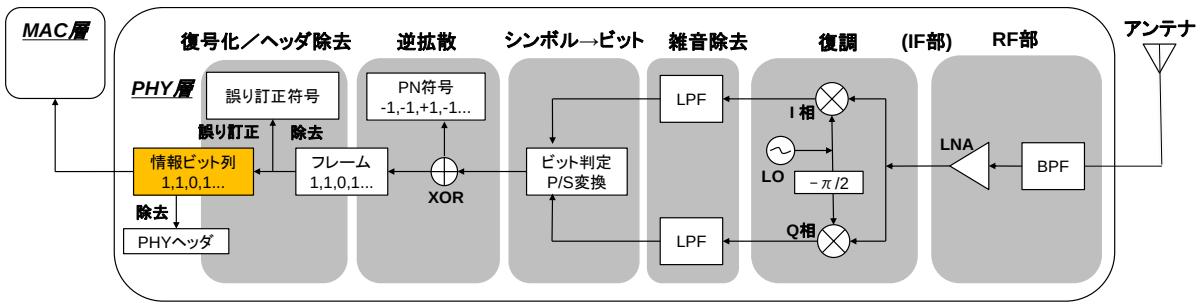


図 3-8 PHY 層における一般的な受信処理の流れ

QualNet の物理層の受信処理では、「PHY ヘッダの除去」までの一切の処理を省略している。これは送信側処理で符号化・拡散・変調処理を実施していない点からすると当然であると言えるが、つまり QualNet では PHY 層において送受信ビット列の各ビットに対する操作やベースバンド信号の生成とその操作、さらに変調波の生成とその操作といった処理は一切省略している。ここで、QualNet では符号化／拡散／変調方式や電波の特性が考慮されないという意味ではなく、プログラム上での送受信ビット列操作やプログラム上での波の生成／合成処理は実施していないという意味であるので、誤解を招かないよう注意しておきたい。

では、「符号化／拡散／変調の具体的な処理を省略しつつ、各種符号化／拡散／変調方式の特性は考慮に入れる」ということを、QualNet はどのようにして実現しているかという点について説明する。

基本的な考え方としては、「QualNet はパケットレベルシミュレータであり、PHY 層から MAC 層へ引き渡す情報ビット列（パケットあるいはフレーム）が受信の都度エラーを含んだものとなるかどうかという点について現実を模擬できれば良い。」ということになる。この模擬の精度が高ければ緻密な無線パケットシミュレーションが実現されることになり、逆に精度が低ければ、大雑把な無線パケットシミュレーションが実現されることになる。どの程度の精度が必要かという点に関してはシミュレーションの実施目的（何を評価したいのか）に依存するため一概には言及できない。なお、一般にこの模擬の精度とシミュレーションの処理負荷（シミュレーションに要する時間）はトレードオフの関係にある。

QualNet では、ほとんどの場合、後述する BER（Bit Error Rate）ベースの受信パケットエラー判定方式を採用している。BER ベース以外の判定方式としては SINR 閾値ベースの判定方式（これも後述する）が使われるケースもあるが、こちらはかなり単純なモデルとなっており大雑把な模擬の例と言ってもよい。いずれの方式においても、前節で説明した受信信号電力（S）と干渉信号電力（I）、雑音電力（N）を使って求められる信号対干渉雑音電力比（SINR：Signal to Interference and Noise power Ratio）をもとにしたエラー判定が行われるので、まずこの SINR の求め方についての説明を行った後、各エラー判定方式を解説する。

3.3.3 SINR（Signal to Interference and Noise power Ratio）の算出

SINR は文字通り、受信信号電力（S）と干渉信号電力（I）、雑音電力（N）の比として求められる指標値である。計算式は以下のとおりである。

$$\mathrm{SINR} = \frac{S\ [\mathrm{mW}]}{\sum_k (I_k\ [mW]) + N[mW]}$$

以下では、受信信号電力（S）、干渉信号電力（I）、雑音電力（N）それぞれに関して、QualNet での計算方法および現実に照らし合わせた想定について述べる。

ただしその前に、受信した電波が希望波（S）と干渉波（I）のいずれかを判定する基準について触れておく。QualNet では、基本的に電波の送信元や宛先で希望波か干渉波かの判定を行っているのではなく、IDLE 状態のときに最初に届いた電波を希望波（S）とみなし、希望波受信中に後から並行して届いた電波を干渉波（I）とみなす。（電波の送信元や宛先情報というのは上位層の概念であって物理層で取り扱うものではないので注意が必要である。）

また以下の閾値が設定可能で、受信電波の大きさが *PHY-ABSTRACT-RX-THRESHOLD* 以下の場合も干渉波（I）とみなされる。

表 3-7 受信電力の閾値

設定項目	設定値	説明
PROPAGATION-LIMIT	dBm	受信アンテナゲインを加算する前の電力がこの値より小さい場合は、電波を受信しない
PHY-ABSTRACT-RX-THRESHOLD	dBm	受信電力がこの値より大きい場合は受信処理を行い、そうでない場合は干渉波として処理する。

3.3.3.1 受信信号電力（S）の算出

QualNet の物理層における受信信号電力（S）は、復調後のベースバンド信号の受信電力として取り扱われる。キャリア電力ではない点に注意する必要がある。

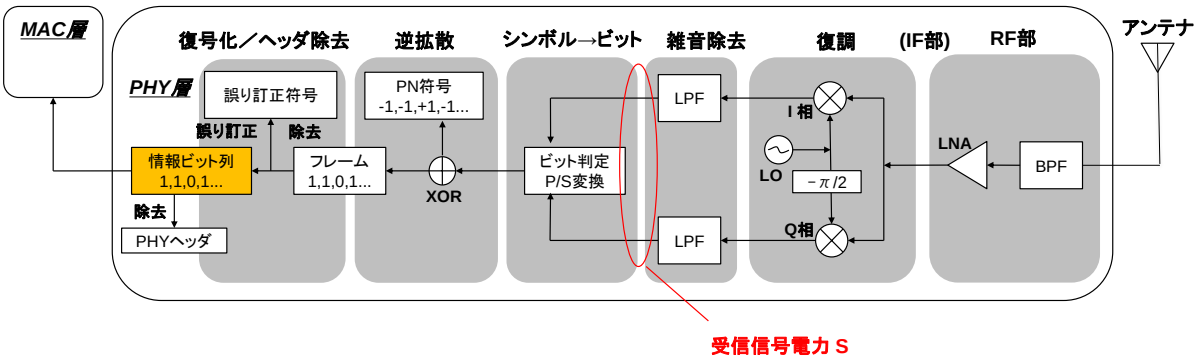


図 3-9 受信信号電力(S)

3.3.3.2 受信干渉電力（I）の算出

干渉電力（I）についても、信号電力（S）と同様に復調後のベースバンド信号の電力として取り扱われる。

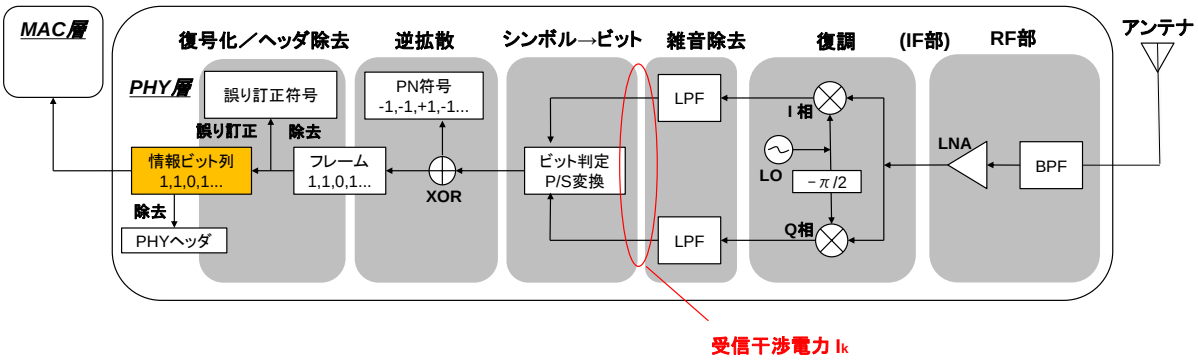


図 3-10 受信干渉電力(I)

また干渉電力については、希望信号（S）と時間軸上で重なって受信したものをすべて重ね合わせた総干渉電力を SINR 計算時に取り扱う。

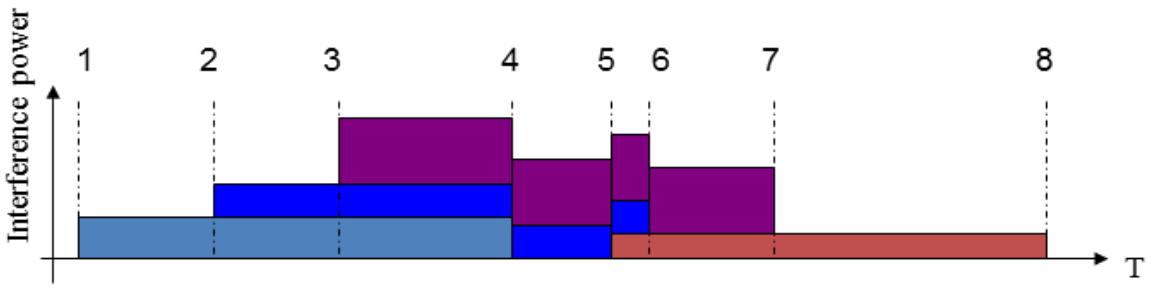


図 3-11 干渉電力の重ね合わせ

3.3.3.3 雑音電力（N）の算出

雑音電力は、前述のとおり AWGN として熱雑音および回路内部雑音が考慮される。

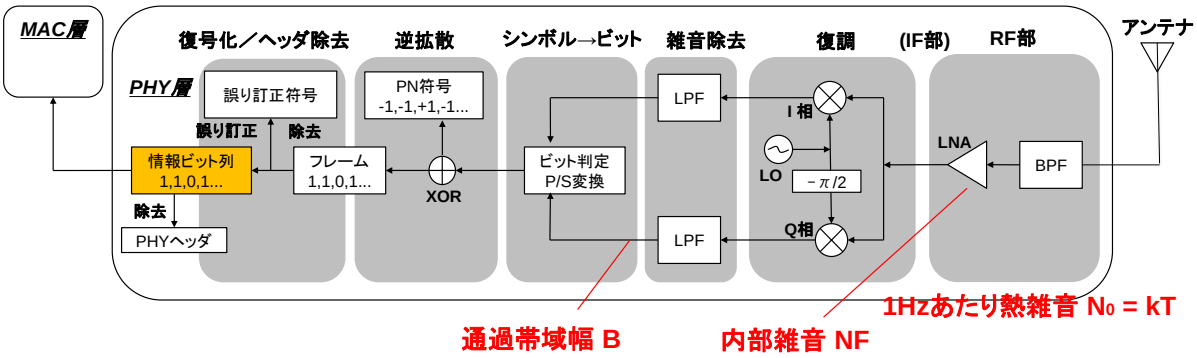


図 3-12　雑音電力(N)

熱雑音については 1Hz あたりの雑音電力にベースバンド受信フィルタの通過帯域幅をかけたものとして算出される。内部雑音としてはノイズファクター（dB 値で表される Noise Figure ではなく真値の値）が設定できるようになっており、トータルの雑音電力は先に示したように下式で計算される値となる。

$$N\ [\mathrm{mW}] = kTB * NF * 1000\ [mW]$$

k：ボルツマン定数（1.38 * 10^-23 [J/K]）
T：温度 [K]
B：ベースバンド受信フィルタの帯域幅 [Hz]
NF：ノイズファクター [dB 値ではなく真値]

3.3.4 受信エラー判定

QualNet では PHY モデルごとに 1 種類以上の受信エラー判定モデルが用意されている。各 PHY モデルが選択可能な受信エラー判定モデルを表 3-8 に示す。

表 3-8　PHY モデルと受信エラー判定モデル

PHY モデル	受信エラー判定モデル	エラー判定方式種別
Abstract PHY	BER-BASED SNR-THRESHOLD-BASED	BER ベース SINR 閾値ベース
802.11 PHY	PHY802.11a PHY802.11b BER-BASED	BER ベース BER ベース BER ベース
802.16 PHY	PHY802.16	BER ベース
802.15.4 PHY	PHY802.15.4 BER-BASED	BER ベース BER ベース
GSM PHY	BER-BASED	BER ベース
UMTS PHY	PHY-UMTS	BER ベース

エラー判定モデルを大別すると、SINR 閾値ベースの判定方式を採用するモデルと BER ベースの判定方式を採用するモデルに分けられる。PHY802.11a モデルなどの PHY プロトコル固有のエラー判定モデルは全て BER ベースのモデルであり、BER 曲線データをシミュレータ内に埋め込みで持っている点だけが BER-BASED モデルと異なる。

以下に、SINR 閾値ベースと BER ベースそれぞれの受信エラー判定方式を説明する。

3.3.4.1 SINR 閾値ベースの受信エラー判定方式

SNR 閾値ベースの受信エラー判定方式では、SINR と設定パラメータで与えられる SNR 閾値を比較し、そのパケットが受信エラーを起こすかどうかを決定する。このように、この方式は SINR に対して確定的なエラー判定方式となっている。

参考までに、Abstract PHY モデルにおいて SNR-THRESHOLD-BASED モデルを選択した際の受信エラー判定処理の実装コードを以下に示す。SINR が閾値未満である場合に必ず packeterror が TRUE に設定されることが確認できる。

phy_abstract.cpp

```
475         if (sinr < phy_abstract->thisPhy->phyRxSnrThreshold) {
476
477             packeterror = TRUE;
478             double interferenceThreshold = 0;
479
480             interferenceThreshold =
481                 phy_abstract->rxMsgPower_mW/
482                 (phy_abstract->thisPhy->phyRxSnrThreshold
483                  - (phy_abstract->thisPhy->noise_mW_hz
484                     * phy_abstract->bandwidth));
485
486             if (phy_abstract->interferencePower_mW > interferenceThreshold)
487             {
488                 phy_abstract->stats.totalSignalsInterferenceErrors++;
489             }
490         } else {
491             packeterror = FALSE;
492         }
```

3.3.4.2 BER ベースの受信エラー判定方式

BER ベースの受信エラー判定方式では、以下の式により BER(Bit Error Rate)から PER(Packet Error Rate)を求めたうえで、乱数を振って出た値を PER と比較し、そのパケットが受信エラーを起こすかどうかを決定する。このように、この方式は SINR に対して確率的なエラー判定方式となっている。

$$PER = 1 - (1 - BER)^{L}$$

where

BER：Bit Error Rate

L　：受信データの総ビット数

参考までに、Abstract PHY モデルにおいて BER-BASED モデルを選択した際の受信エラー判定処理の実装コードを以下に示す。BER は後述するように、PHY_BER 関数を呼び出して取得する。PER はここでは errorProbability という変数で表現されている。

phy_abstract.cpp

```
497         BER = PHY_BER(phy_abstract->thisPhy,
498                       0,
499                       sinr);
500
501         if (BER != 0.0) {
502             double numBits =
503                 ((double)(getSimTime(node) - phy_abstract->rxTimeEvaluated)
504                  * (double)phy_abstract->dataRate /(double)SECOND);
505
506
507             double errorProbability = 1.0 - pow((1.0 - BER), numBits);
```

```
508             double rand = RANDOM_erand(phy_abstract->thisPhy->seed);
509
510             assert((errorProbability >= 0.0) && (errorProbability <= 1.0));
511
512             if (errorProbability > rand) {
513                 packeterror = TRUE;
514             }else{
515                 packeterror = FALSE;
516             } //if errorProbability
517         } // if BER
```

3.3.5 BER (Bit Error Rate)

3.3.5.1 BER 計算

QualNet での BER(Bit Error Rate)の算出は、BER 計算用 API である PHY_BER 関数を呼び出すことで行われる。PHY_BER 関数の呼び出し形式を以下に示す。

phy.h

```
962 // /**
963 // API             :: PHY_BER
964 // LAYER           :: Physical
965 // PURPOSE         :: Get BER
966 // PARAMETERS      ::
967 // + phyData        : PhyData * : PHY layer data
968 // + berTableIndex : int        : index for BER tables
969 // + sinr           : double     : Signal to Interference and Noise Ratio
970 // RETURN          :: double     : Bit Error Rate
971 // **/
972 double PHY_BER(
973     PhyData *phyData,
974     int berTableIndex,
975     double sinr);
```

PHY_BER 関数の実装コードは公開されていないが、BER 曲線データ(SINR vs. BER データ)の線形補間により、第 3 引数で与えられた SINR 値に対応する BER 値が算出されるようになっている。BER 曲線データの定義域外の補間に関しては、図 3-13 に示すように下限 SINR 値あるいは値 0 での補間となる。なお、プログラム内で取り扱われる SINR 値は真値であるが、図 3-13 の横軸の SINR 値はグラフの見やすさのために dB 換算値を用いているので注意が必要である。

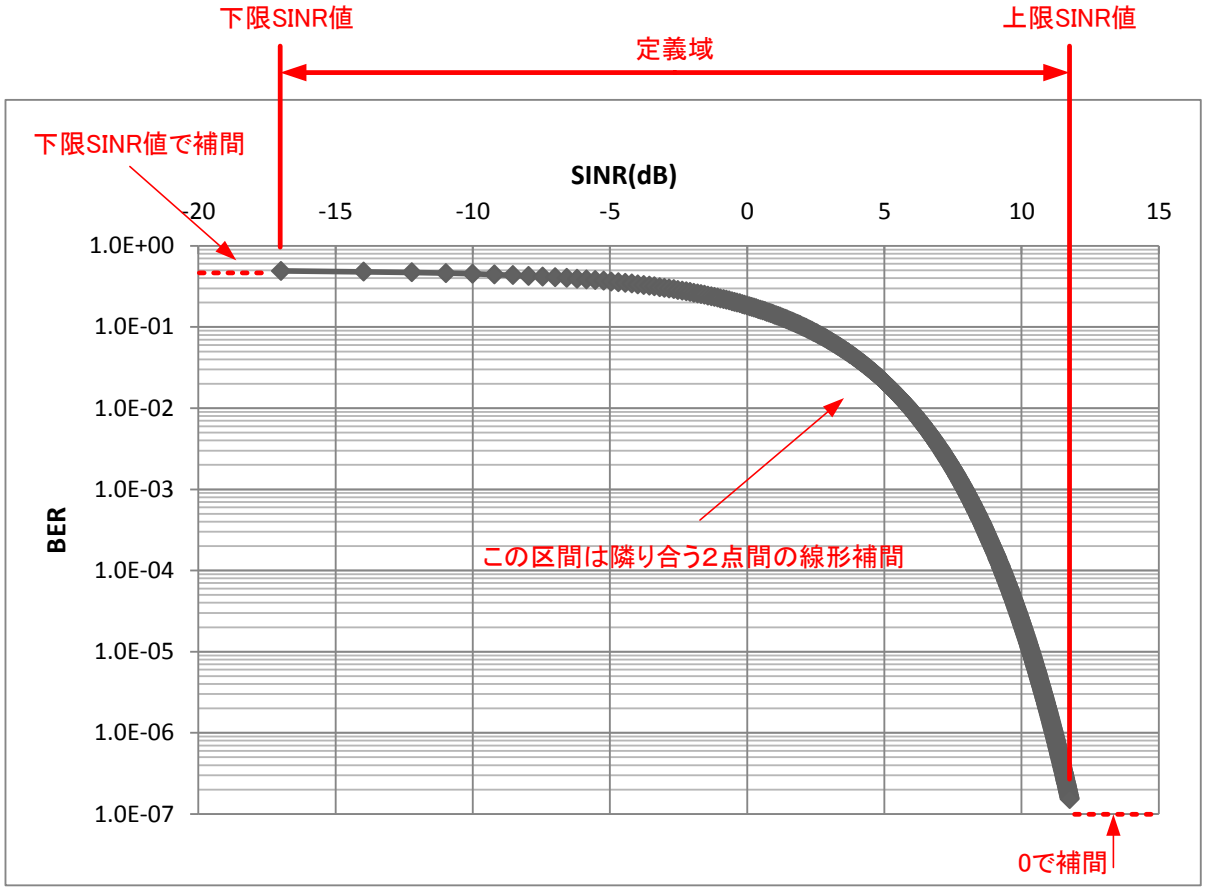


図 3-13　BER 曲線データの補間(横軸は dB 換算値)

BER 曲線データ自体は、受信エラー判定モデルが BER-BASED モデルの場合には設定パラメータとして外部からファイルで与えるが、その他の BER ベースの判定方式を用いるモデルの場合には QualNet カーネル内部に埋め込まれており外部から与えることができない。ただしいずれの場合も、図 3-14 に示すように PhyData 構造体内の snrBerTables が指すメモリ領域にデータは格納されるので、曲線データを参照することは可能である。なお、図中の PhyBerTable 構造体における isFixedInterval から snrEnd までの要素は、BER 曲線データの SNR 値(真値)が一定間隔で並んでいる場合にのみ値が設定される要素であり、計算速度向上のための仕掛けと思われる。

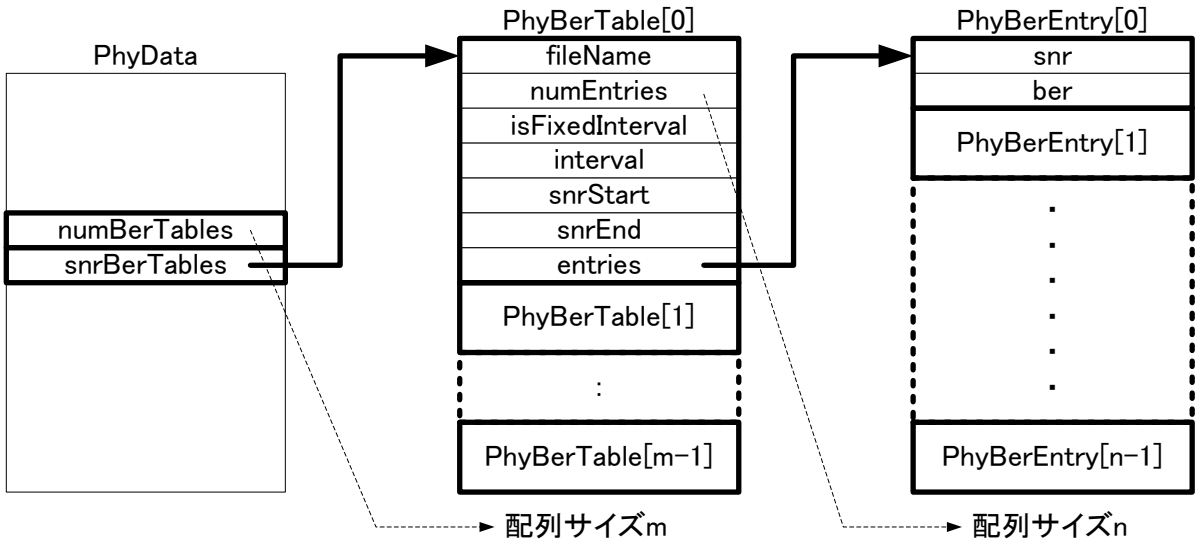


図 3-14 BER 曲線データ格納構造

また、PhyData 構造体には、BER 曲線データと同じ格納構造で、snrPerTables と snrSerTables という要素があり、PER(Packet Error Rate)曲線データと SER(Symbol Error Rate)曲線データが格納できるようになっている。ただし、Version5.1 の時点ではこれらのデータを利用しているコードは見当たらなかったので、将来そのような受信エラー判定モデルが追加される可能性が窺える。

3.3.5.2 BER 曲線データ

これまで解説から分かるように、QualNet における物理層モデルの特性は、BER 曲線データに集約されていると言える。そのため、BER 曲線データが想定と異なる場合には全く想定外のシミュレーション結果が得られることも考えられるため、無線シミュレーションを行う際には使用する BER 曲線データがどのような特性を持っているかを十分に理解しておいたほうが良い。

参考までに、以下にいくつかの受信エラー判定モデルの BER 曲線データを掲載する。

まず BER-BASED モデルの場合の例を挙げる。この場合、BER 曲線データは前述のように外部ファイルで与えるが、QualNet ではに示す外部ファイルが QUALNET_HOME/data/modulation ディレクトリに標準で用意されておりこれらを指定することができる。もちろんこれら以外にユーザが自身で作成した BER 曲線データを使用することも可能である。BER 曲線データファイルは、各行に SINR 値(真値)と対応する BER 値をスペース区切りで並べたテキスト形式ファイルである。

表 3-9 標準で用意されている BER 曲線データファイル

File	Description
bpsk.ber	Binary phase-shift keying modulation. No encoding.
bpsk-turbo.ber	Binary phase-shift keying modulation with turbo encoding.
cck-5_5.ber.ber	Complimentary code keying for 5.5 Mbps.
cck-11.ber	Complimentary code keying for 11 Mbps.
dpsk.ber	Differential phase-shift keying modulation. No encoding.
dpsk-turbo.ber	Differential phase-shift keying modulation with turbo encoding.
dqpsk.ber	Differential quadrature phase-shift keying modulation. No encoding.
fsk2.ber	Binary Frequency-shift keying modulation. No encoding. May be used for analog data up to 16 kbits/s.
fsk8.ber	M-ary Frequency-shift keying modulation, M=8. No encoding. Provides support for ALE.
fsk8_golay.ber	M-ary Frequency-shift keying modulation, M=8 with Golay encoding. Provides support for ALE.
gmsk.ber	Gaussian minimum shift keying modulation. No encoding.

qam64.ber	64-Quadrature amplitude modulation. No encoding. May be used for the cases that have a 27Mbps capability over a 6MHz line.
qam64-convolutionalr12.ber	64-Quadrature amplitude modulation with convolutional encoding, with code rate 1/2. Provides the ability to use FEC.
qam64-convolutionalr23.ber	64-Quadrature amplitude modulation with convolutional encoding, with code rate 2/3. Provides the ability to use FEC.

これらのデータをプロットしたグラフを図 3-15 に示す。

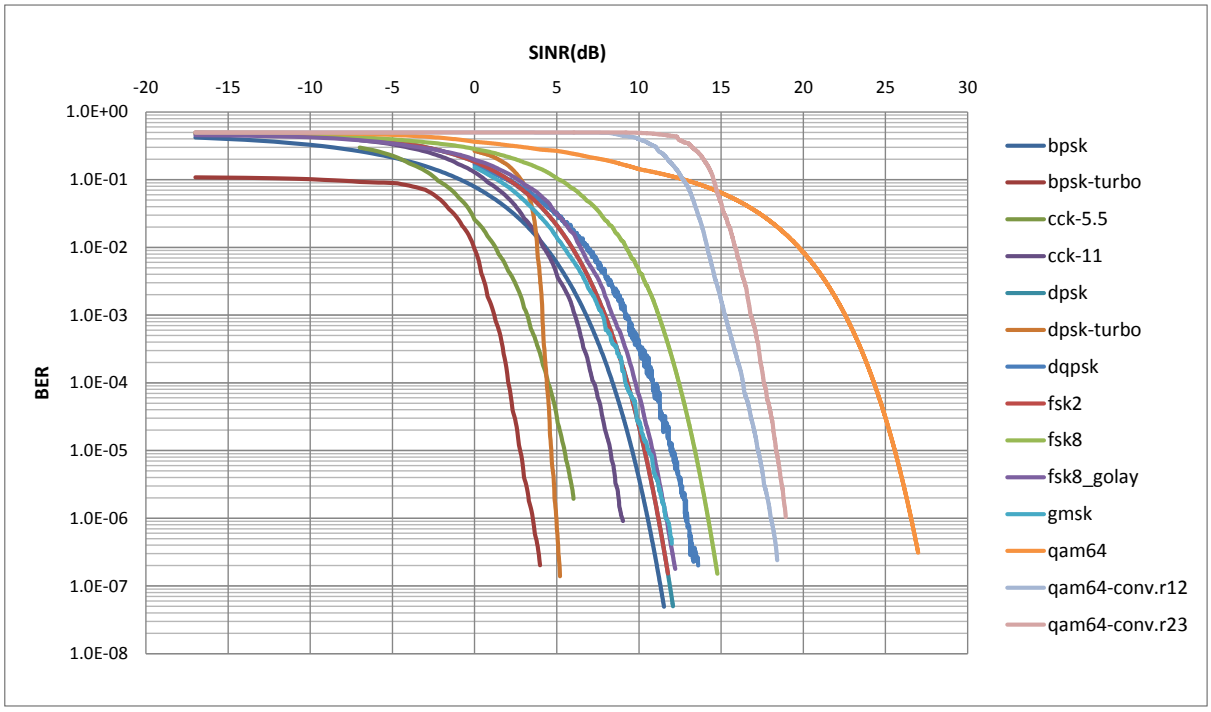


図 3-15　標準で用意されている BER 曲線データ(横軸は dB 換算値)

次に PHY802.11a モデル及び PHY802.11b モデルの場合の例を挙げる。これらの場合は、BER 曲線データは前述のように QualNet カーネル内部に格納されている。PHY_CreateAPhyForMac 関数の末尾に以下の printf 文を追加して QualNet を動作させるとこの曲線データの値が取得できる。その方法で取得したデータをプロットしたグラフを図 3-16 と図 3-17 に示す。

phy.cpp

```
for (i = 0; i < thisPhy->numBerTables; i++) {
       struct PhyBerTable *phyBerTable = &(thisPhy->snrBerTables[i]);
       printf("BER Table[%d] %d Entries\n", i, phyBerTable->numEntries);
       printf("snrStart %lf snrEnd %lf\n",
              hyBerTable->snrStart, phyBerTable->snrEnd);
       for (int j = 0; j < phyBerTable->numEntries; j++)
       {
              printf("%lf,%lf\n",
                     phyBerTable->entries[j].snr,
                     phyBerTable->entries[j].ber);
       }
}
```

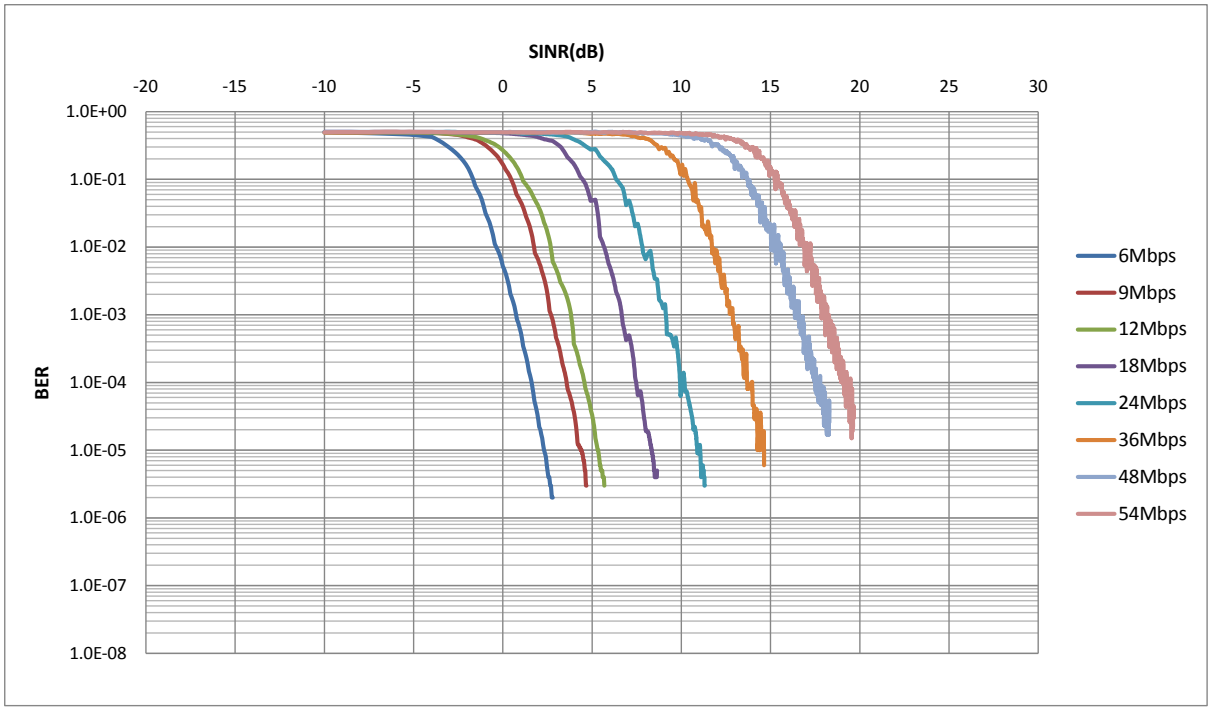


図 3-16　PHY802.11a モデルの BER 曲線データ(横軸は dB 換算値)

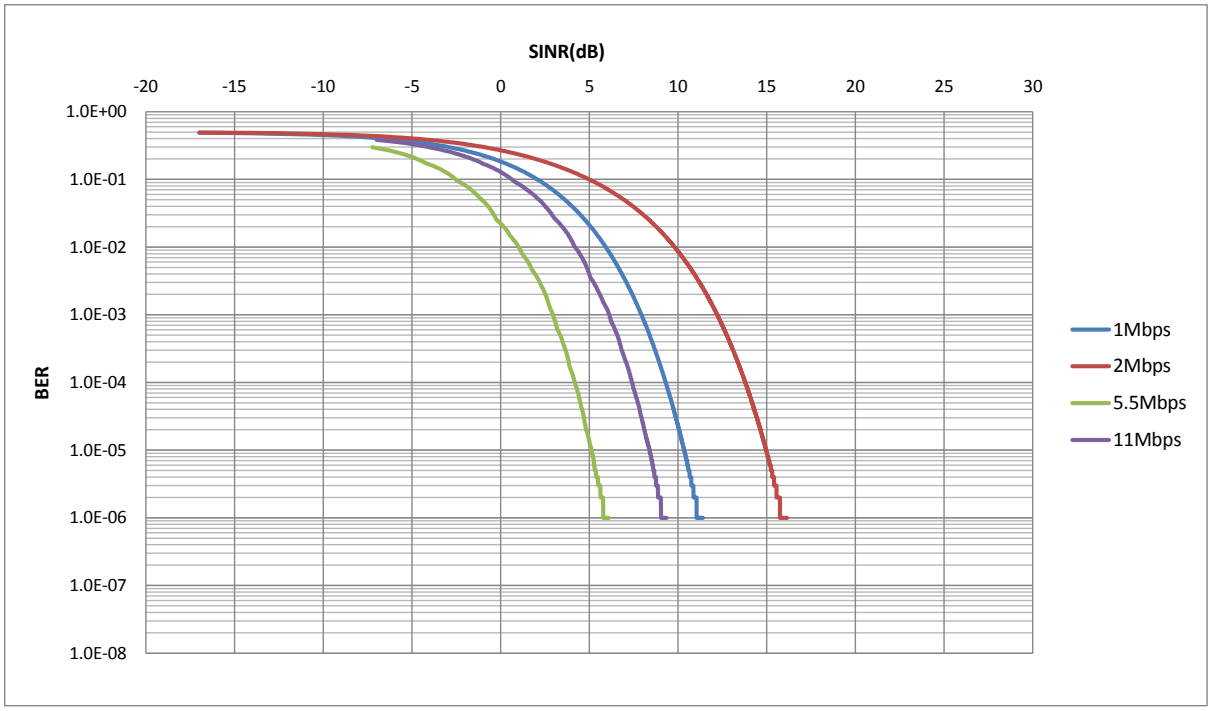


図 3-17　PHY802.11b モデルの BER 曲線データ(横軸は dB 換算値)

PHY802.11b モデルの場合、5.5Mbps 及び 11Mbps の場合の BER 特性が 1Mbps 及び 2Mbps の場合の BER 特性よりも良いというデータが得られているが、これは 1Mbps/2Mbps の場合と 5.5Mbps/11Mbps の場合でカーネル内部に格納されている BER 曲線データのデータ作成時の前提条件が異なるためである。PHY802.11b モデルでは、伝送モード(1Mbps/2Mbps/5.5Mbps/11Mbps)にかかわらず、本来 IEEE802.11b の 1Mbps/2Mbps モードで使用される DSSS 変復調方式の拡散利得を考慮した信号電力と雑音電力が算出される。DSSS 方式では 11 チップの Barker 符号で拡散・逆拡散するので拡散利得は 11 倍(10.41dB)となる。つまり受信側での雑音は 11 分の 1 倍に拡

散されて算出される。1Mbps/2Mbps の BER 曲線データはこの DSSS による拡散利得を考慮して計算された SINR に対する BER 特性となっているが、一方で 5.5Mbps/11MbpsBER 曲線データは DSSS 変復調方式ではなく CCK 変復調方式の特性データであり、DSSS による拡散利得を考慮せずに計算された SINR に対する BER 特性である。そのため、Phy802_11CheckRxPacketError 関数において、PHY_BER 関数を呼び出す直前で受信伝送速度をチェックし、5.5Mbps と 11Mbps の場合は雑音電力を 11 倍することで BER 計算時の入力 SINR から現拡散利得分を差し引いていることが確認できる。(なお補足になるが、定数 PHY802_11b__6M は定数名に"6M"と書かれているが、5.5Mbps の伝送速度種別を表す定数である。)

phy_802_11.cpp

```
172 BOOL Phy802_11CheckRxPacketError(
173     Node* node,
174     PhyData802_11* phy802_11,
175     double *sinrPtr)
176 {
177     double sinr;
178     double BER;
179     double noise =
180         phy802_11->thisPhy->noise_mW_hz * phy802_11->channelBandwidth;
        ... 中略 ...
184     if (phy802_11->thisPhy->phyModel == PHY802_11b &&
185         (phy802_11->rxDataRateType == PHY802_11b__6M ||
186          phy802_11->rxDataRateType == PHY802_11b_11M))
187     {
188         noise = noise * 11.0;
189     }
190
191     sinr = (phy802_11->rxMsgPower_mW /
192             (phy802_11->interferencePower_mW + noise));
        ... 中略 ...
202     BER = PHY_BER(phy802_11->thisPhy,
203                   phy802_11->rxDataRateType,
204                   sinr);
        ... 中略 ...
223 }
```

よって例えば、図 3-18 のように 5.5Mbps と 11Mbps の場合の BER カーブの横軸の値を 11 倍した(約 10dB 右へずらした)場合の特性グラフが、実際の特性に近いイメージであると言える。

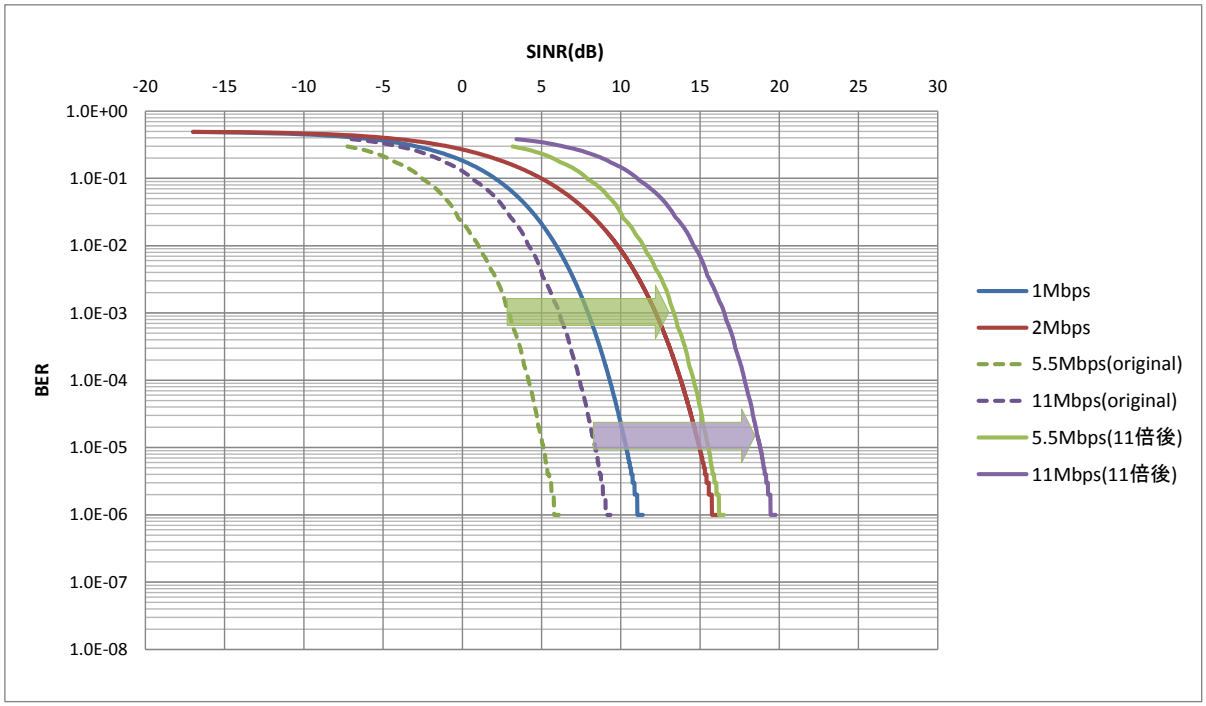


図 3-18　PHY802.11b モデルの BER 特性イメージ(横軸は dB 換算値)

3.4 アンテナモデル

送受信アンテナゲイン(G_T,G_R)は指定したアンテナモデルによって決まる。アンテナモデルは任意のアンテナモデルもしくは、組込のアンテナモデルを使用できる。どちらを使用するかは、*ANTENNA-MODEL-CONFIG-FILE-SPECIFY* を YES(任意のモデル)または NO(組込のモデル)と指定することで選択可能である。

3.4.1 任意のアンテナモデル

ANTENNA-MODEL-CONFIG-FILE で設定したファイルでは Open-ASCII (2-D/3-D) や NSMA formats といったモデルや、ユーザ定義によるモデルを *ANTENNA-PATTERN-TYPE* で設定可能である。その中で実際に使用するモデルを *ANTENNA-MODEL* で指定可能である。

使用時の詳細は「QualNet-5.1-Wireless-ModelLibrary.pdf」の「Patterned Antenna Model」の章を参照のこと。

3.4.2 組込のアンテナモデル

ANTENNA-MODEL に以下の設定を行うことで、組込のアンテナモデルを使用することができる。

表 3-10 ANTENNA-MODEL

モデル名	設定値
Omnidirectional	OMNIDIRECTIONAL
Switched Beam	SWITCHED-BEAM
Steerable	STEERABLE

各モデルの説明を以下に記す。

3.4.2.1 Omnidirectional

全ての到来角度、送信角度に対して、同じゲイン値を返す無指向のモデルである。本モデルで は 、ANTENNA-EFFICIENCY、ANTENNA-MISMATCH-LOSS、ANTENNA-CABLE-LOSS、ANTENNA-CONNECTION-LOSS を設定するほか、*ANTENNA-GAIN* でゲイン[dB]を設定し、その値が送受信アンテナゲイン(G_T,G_R)となる。

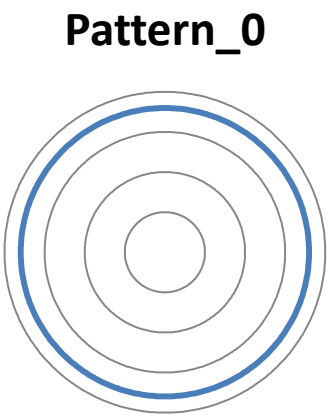


図 3-19 Omnidirectional のパターン

3.4.2.2 Switched Beam

複数方向に輻射パターンを(シミュレーションパラメータとして)定義できるモデルである。送受信信号ごとにどのパターンを使うかを、動的に(プログラム中で)指定することが可能である。

受信時に与えられた到来角度、信号に対して最も高いゲインを得られるパターンを返す。8 つのゲインパターンが異なる方角に対して定義される。

以下に示すアンテナパターンは、 $QUALNET_HOME/data/antenna/default.antenna-azimuth を図化したものである。

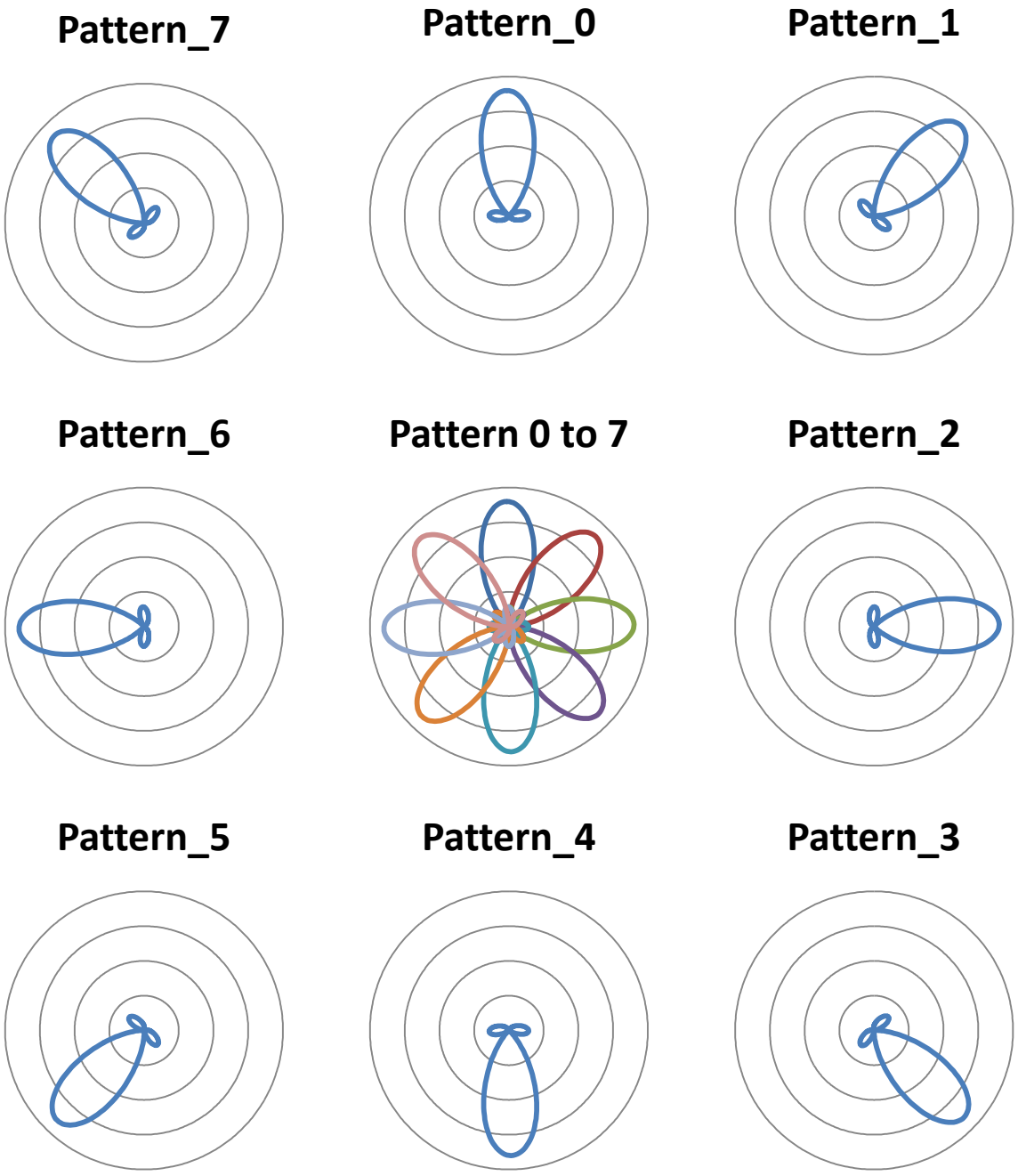


図 3-20 Switched Beam のパターン

本モデルでは、ANTENNA-EFFICIENCY、ANTENNA-MISMATCH-LOSS、ANTENNA-CABLE-LOSS、ANTENNA-CONNECTION-LOSS を設定するほか、*ANTENNA-GAIN* でゲイン[dB]を設定する。

ANTENNA-AZIMUTH-PATTERN-FILE と *ANTENNA-ELEVATION-PATTERN-FILE* では水平、垂直方向のパターンを設定できる。

ファイルの詳細に関しては、「QualNet-5.1-Wireless-ModelLibrary.pdf」の「Switched-beam Antenna Model」を参照のこと。

その結果が送受信アンテナゲイン(G_T,G_R)となる。

3.4.2.3 Steerable

複数方向に異なるビーム幅のパターンを定義できるモデルである。送受信信号ごとにどのパターンを使うかを動的に(プログラム中で)指定できる。

受信信号に対して、最も高いゲイン値の得られる方向に回転させることができます。サイドローブを含めた 10 パターンが定義できる。

以下に示すアンテナパターンは、 $QUALNET_HOME/data/antenna/steerable.antenna-azimuth を図化したものである。

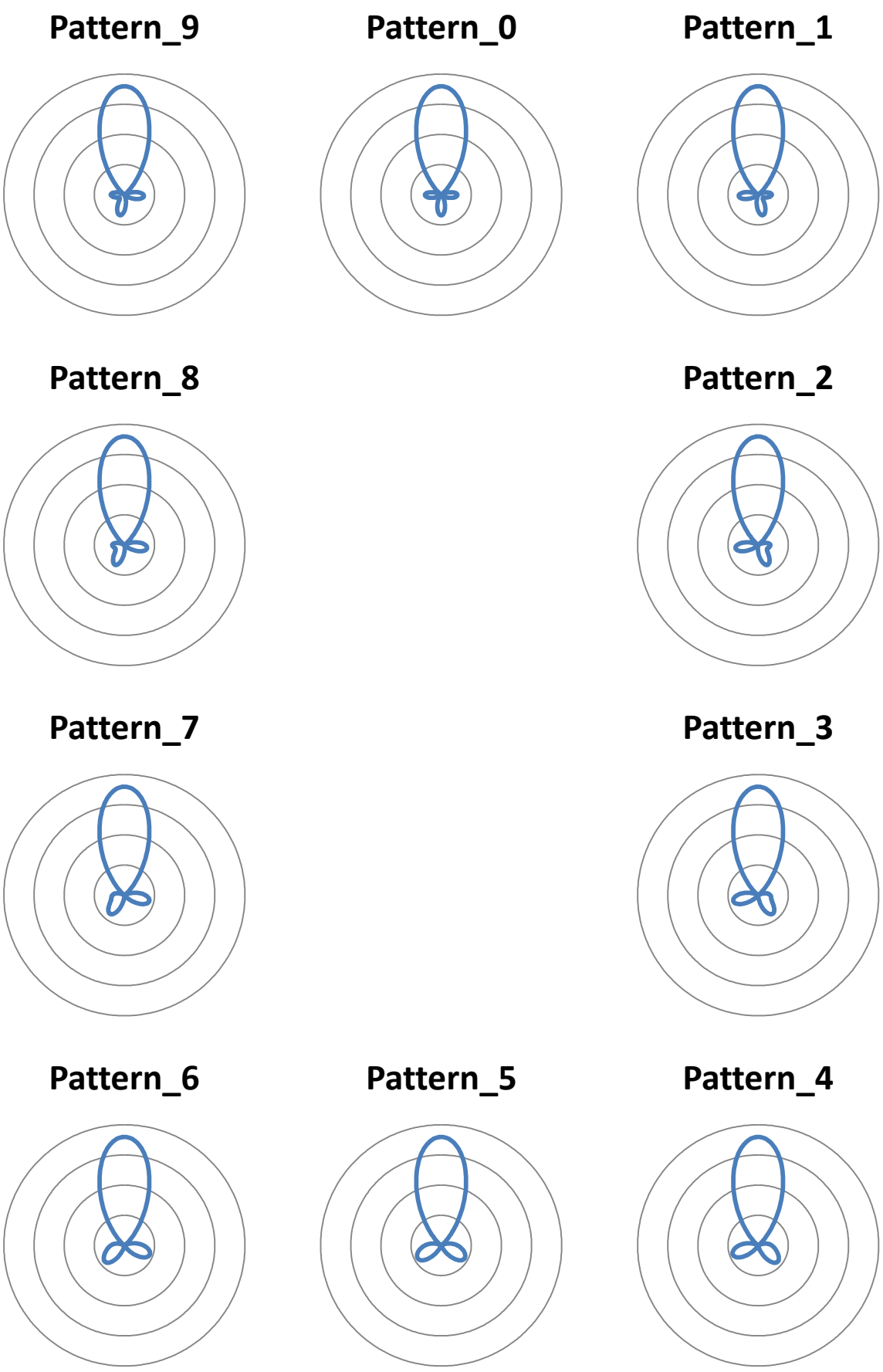


図 3-21　Steerable のパターン

本モデルでは、ANTENNA-EFFICIENCY、ANTENNA-MISMATCH-LOSS、ANTENNA-CABLE-LOSS、ANTENNA-CONNECTION-LOSS を設定するほか、*ANTENNA-GAIN* でゲイン[dB]を設定する。

ANTENNA-AZIMUTH-PATTERN-FILE と *ANTENNA-ELEVATION-PATTERN-FILE* で水平、垂直方向のパターンを設定できる。

ファイルの詳細に関しては、「QualNet-5.1-Wireless-ModelLibrary.pdf」の「Steerable Antenna Model」を参照のこと。

その結果が送受信アンテナゲイン(G_T,G_R)となる。

3.4.2.4 asciiazimuth

$QUALNET_HOME/data/には、default.antenna-azimuth や steerable.antenna-azimuth だけでなく、default.asciiazimuth も入っているが、この default.asciiazimuth で定義されているデータは以下に示すようにかなり荒い。

```
#Example for 2D ASCII pattern for Azimuth.
#Angle [degree]    Gain[dB]
#Number of samples in the file
13

0.0                -4.0
4.0                 0.0
10.0               -3.0
15.0              -10.0
20.0              -20.0
90.0              -20.0
180.0             -20.0
270.0             -20.0
340.0             -20.0
345.0             -10.0
350.0              -3.0
356.0               0.0
360.0               0.0
```

以下に示すアンテナパターンは、この default.asciiazimuth を図化したものである。

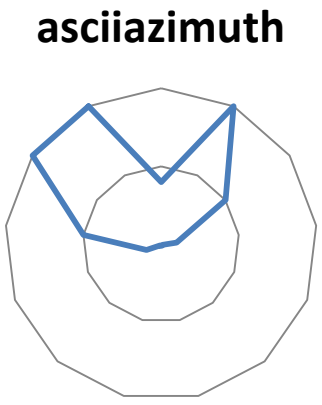


図 3-22 default.asciiazimuth

アンテナパターンが非対称になっているが、定義データの最後の一行を下記のように変更することで対称なアンテナパターンになる。

```
360.0                -4.0
```

asciiazimuth'

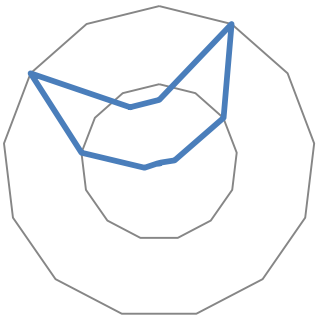


図 3-23　補正 default.asciiazimuth

参考までに、3 次元アンテナパターンが定義されている default.ascii3d も図化する。

```
#Theta is the elevation angle.
#Phi is the azimuth angle.
#a(Theta,Phi) [dB] is the Gain in dB.


# Theta             Phi             a(Theta,Phi) [dB]
0                   0                  -20.0
0                   90                 -17.0
0                   180               -12.0
0                   270                -17.0
0                   360                -20.0
90                  0                   5.0
90                  90                  0.0
90                  180                -25.0
90                  270                 0.0
90                  360                 5.0
180                 0                  -20.0
180                 90                 -17.0
180                 180                12.0
180                 270                -17.0
180                 360                -20.0
```

この定義データを見てわかるように、default.asciiazimuth 同様に 3 次元の default.ascii3d データも定義データ数が少なく、単純に 3D 表示するとこのような図になる。

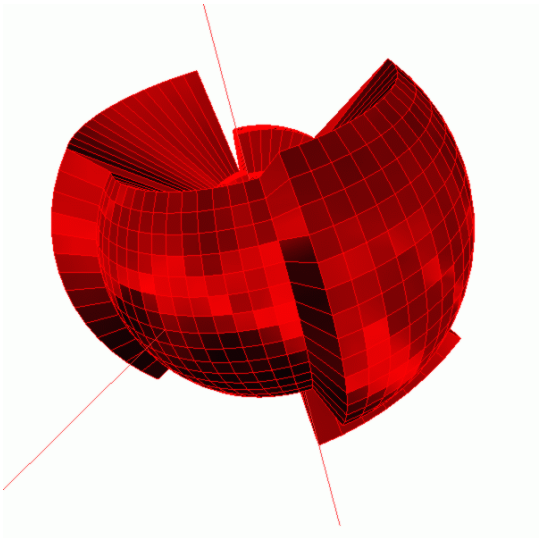


図 3-24　default.ascii3d

3.5 無線伝搬路モデル

QualNet では伝搬路における受信電界の減衰特性を以下の独立した特性に分解してそれぞれモデル化している。

・長区間変動（Pathloss：距離減衰ならびに物体反射・透過減衰の影響）
・短区間変動（Shadowing：障害物等による遮蔽の影響）
・瞬時変動（Fading：マルチパスフェージングとドップラー周波数偏移の影響）
・気象変動（降雨量の影響）

伝搬路全体で受ける減衰量はこれらの足し合わせにより得られる。以下、それぞれに関する説明を記す。なお気象変動については、Weather オブジェクトを設定することで考慮に入れることが可能となっているという説明に留める。降雨減衰モデルは以下のドキュメントを参考にして実装されている。

ITR-R Recommendation P.838-1. “SPECIFIC ATTENUATION MODEL FOR RAIN FOR USE IN PREDICTION METHODS”, Geneva, 2000.

3.5.1 Pathloss

Pathloss のモデルは、*PROPAGATION-PATHLOSS-MODEL* で設定する。

標準で提供されるモデルもあれば、有償ライブラリとして提供されるモデルや一般には提供しないモデルもあるので、注意が必要である。

表 3-11 標準で提供されるモデル (Wireless Library)

モデル名	設定値
Free Space	FREE-SPACE
Two Ray	TWO-RAY
Pathloss Matrix	PATHLOSS-MATRIX
Irregular Terrain Model	ITM

表 3-12 有償ライブラリで提供されるモデル (Urban Propagation Library)

モデル名	設定値
Okumura-Hata Propagation Model	OKUMURA-HATA
COST 231-Hata Propagation Model	COST231-HATA
COST 231-Walfish-Ikegami (COST-WI) Propagation Model	COST231-WALFISH-IKEGAMI
Automatic Model Selection	URBAN-MODEL-AUTOSELECT
Street Microcell Propagation Model	STREET-MICROCELL
Street Mobile-to-mobile Propagation Model	STREET-M-TO-M
Suburban Propagation Model	SUBURBAN

表 3-13 一般には提供しないモデル

モデル名	設定値
OPNET Path Attenuation Routine	OPAR
Terrain Integrated Rough Earth Model	TIREM
Advanced Stand Alone Prediction Service Propagation Model	ASAPS

OPNET Path Attenuation Routine	PATHLOSS-OPAR
OPNET Path Attenuation Routine	PATHLOSS-OPAR-PROP
不明	RFPS

本書では標準で提供されるモデルと有償ライブラリで提供されるモデルを説明する。

なお Pathloss は送受信点の位置が変わらない場合は、直近の計算結果を再利用し再計算は行われない。

libraries/wireless/src/propagation.cpp の 2124 行目から Pathloss モデルのパラメータ読み込みを行う。

propagation.cpp

```
02125         // Set pathlossModel
02126         //
02127         IO_ReadStringInstance(
02128             ANY_NODEID,
02129             ANY_ADDRESS,
02130             nodeInput,
02131             "PROPAGATION-PATHLOSS-MODEL",
02132             channelIndex,
02133             TRUE,
02134             &wasFound,
02135             buf);
02136
02137         if (wasFound) {
02138             if (strcmp(buf, "FREE-SPACE") == 0) {
02139                 propProfile->pathlossModel = FREE_SPACE;
02140             }
02141             else if (strcmp(buf, "TWO-RAY") == 0) {
02142                 propProfile->pathlossModel = TWO_RAY;
02143             }
…
```

行番号	処理内容
2131	PROPAGATION-PATHLOSS-MODEL パラメータの値を buf で取り出し、
2138	各モデルの設定値(” FREE-SPACE”や” TWO-RAY”など)をチェックして、該当のモデルを propProfile->pathlossModel に設定する。 ここで設定されるのは include/propagation.h で定義される enum PathlossModel である。

3.5.1.1 Free Space

Friis の自由空間損失モデル。パスロス計算式は以下に従う。

$$P_{PATHLOSS} = 20 * \log_{10}(4.0 * D * \lambda)$$

D：送受信点間距離[m]

λ：波長[m]

QualNet では光速を $3*10^8$ [m/sec]と定義しており、

$3*10^8$ / *PROPAGATION-CHANNEL-FREQUENCY* で波長が求まる。

PROPAGATION-CHANNEL-FREQUENCY に関しては、「0」を参照のこと。

パスロス計算の実装箇所を以下に示す。

propagation.cpp

```
00113 double PROP_PathlossFreeSpace(double distance,
00114                               double waveLength)
00115 {
00116     double pathloss_dB = 0.0;
```

```
00117     double valueForLog = 4.0 * PI * distance / waveLength;
00118
00119     if (valueForLog > 1.0) {
00120         pathloss_dB = 20.0 * log10(valueForLog);
00121     }
00122
00123     return pathloss_dB;
00124 }
```

行番号	処理内容
117-120	上記計算式の真数部分を計算してから、式全体を計算する。
119	パスロスが負になる場合は 116 行で設定した 0.0 となる。

3.5.1.2 Two Ray

大地反射の 2 波モデル。自由空間損失と大地反射による損失のうち、損失が大きい結果を採用する。

パスロス計算式は以下に従う。

$$P_{PATHLOSS\_FREESPACE} = 20 * \log_{10}(4.0 * D * \lambda)$$

$$P_{PATHLOSS\_EARTH} = 20 * \log_{10}\left(\frac{D * D}{H_T * H_R}\right)$$

$$P_{PATHLOSS} = \max(P_{PATHLOSS\_FREESPACE}, P_{PATHLOSS\_EARTH})$$

D：送受信点間距離[m]
λ：波長[m]
QualNet では光速を $3*10^8$ [m/sec]と定義しており、
$3*10^8$ / *PROPAGATION-CHANNEL-FREQUENCY* で波長が求まる。
PROPAGATION-CHANNEL-FREQUENCY に関しては、「0」を参照のこと。
H_T：ANTENNA-HEIGHT で指定した送信アンテナ高 [m]
H_R：ANTENNA-HEIGHT で指定した受信アンテナ高 [m]

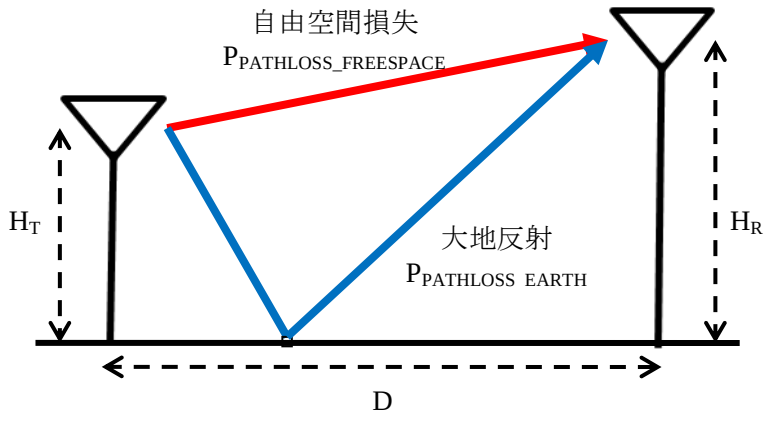


図 3-25 Two Ray モデル

パスロス計算の実装箇所を以下に示す。

propagation.cpp

```
00086 double PROP_PathlossTwoRay(double distance,
00087                            double waveLength,
00088                            float txAntennaHeight,
```

```
00089                             float rxAntennaHeight)
00090 {
00091     double pathloss_dB = 0.0;
00092     double valueForPlaneEarthLoss;
00093     double valueForFreeSpaceLoss;
00094
00095     valueForPlaneEarthLoss =
00096         distance * distance / (txAntennaHeight * rxAntennaHeight);
00097
00098     valueForFreeSpaceLoss = 4.0 * PI * distance / waveLength;
00099
00100     if (valueForPlaneEarthLoss > valueForFreeSpaceLoss) {
00101         if (valueForPlaneEarthLoss > 1.0) {
00102             pathloss_dB = 20.0 * log10(valueForPlaneEarthLoss);
00103         }
00104     } else {
00105         if (valueForFreeSpaceLoss > 1.0) {
00106             pathloss_dB = 20.0 * log10(valueForFreeSpaceLoss);
00107         }
00108     }
00109
00110     return pathloss_dB;
00111 }
```

行番号	処理内容
95	大地反射モデルの対数の真数部分を計算する。
98	自由空間損失の対数の真数部分を計算する。
100-108	真数部分を比較し、採用するモデルを選択する。パスロスが負になる場合は 91 行目で設定した 0.0 となる。

3.5.1.3 Irregular Terrain Model

地形を考慮したパスロス計算のモデル。見通しのある環境では Freespace で計算を行い、見通しが無い場合は、地形(曲率など)・表面情報(伝導率・誘電率)・湿度・気候・偏波を元にパスロス計算を行う。本モデルは Longley-Rice model の改良型であり、広大な土地での適用に向いている。周波数は 20MHz から 20GHz が適用範囲である。

3.5.1.4 Pathloss Matrix

予め送受信ノードとパスロス値を設定したファイルから読み込むモデル。
詳細は「QualNet-5.1-Wireless-ModelLibrary.pdf」の P37 「2.6 Pathloss Matrix Model」を参照。

パスロスマトリックスファイルのフォーマットは以下の通り。
Freq:<Num-channels>:<Frequency 1>:<Frequency 2> ...:<Frequency n>
Nodes:<Num-nodes>
<Time> <Node 1> <Node 2> <Pathloss 1> <Pathloss 2>... <Pathloss m>
<Time> <Node 1> <Node 3> <Pathloss 1> <Pathloss 2>... <Pathloss m>
<Time> <Node 1> <Node 4> <Pathloss 1> <Pathloss 2>... <Pathloss m>

<Num-channels>は、設定する周波数の数
<Frequency i>は、周波数(GHz 単位)
<Num-nodes>は、ノードの数
<Time>は、シナリオ上の経過時間(シミュレーション時間)
<Node i>は、ノードの ID(シナリオ上の番号)
<Pathloss i>はパスロスの値(dB)
※行の順序は問わない。ノード ID 順でもシミュレーション時間でなくともよい。

周波数の数に対してパスロス値の値が少ない場合は最後のパスロス値が使用される。

記述例)
Freq:2:2.4:2.6
Nodes:6
#Time Node1 Node2 Pathloss1 Pathloss2
0 1 2 95.2400000
0 1 3 300.1510000
0 2 1 295.2400000 100.0000000
10 1 2 195.2400000 5.2400000
10 1 3 320.1510000 3.1510000
10 2 3 196.7650000 6.7650000

パスロス計算の実装箇所を以下に示す。

prop_plmatrix.cpp

```
00352 double PathlossMatrix(
00353     Node* node,
00354     NodeAddress nodeId1,
00355     NodeAddress nodeId2,
00356     int channelIndex,
00357     clocktype currentTime)
00358 {
00359     PartitionData* partitionData = node->partitionData;
00360     PropChannel* propChannel = node->partitionData->propChannel;
00361     PropProfile* propProfile0 = propChannel[0].profile;
00362
00363     pair <NodeId, NodeId> srcdstPair (nodeId1, nodeId2);
00364     map < pair<NodeId, NodeId>, double> ::iterator it;
00365
00366
00367     if (partitionData->plNextLoadTime < currentTime) {
00368         PathlossMatrixUpdate(
00369             node->partitionData,
00370             propChannel,
00371             propProfile0->numChannelsInMatrix,
00372             getSimTime(node));
00373     }
00374
00375     it = partitionData->pathLossMatrix[channelIndex]->
00376                             find(srcdstPair);
00377
00378     if (it == partitionData->pathLossMatrix[channelIndex]->
00379                             end())
00380     {
00381        srcdstPair = make_pair(nodeId2, nodeId1);
00382
00383         it = partitionData->pathLossMatrix[channelIndex]->
00384                             find(srcdstPair);
00385
00386         if (it == partitionData->pathLossMatrix[channelIndex]->
00387                             end())
00388         {
00389             return HIGH_PATHLOSS;
00390         }
00391         else
00392         {
00393             return it->second;
00394         }
00395     }
00396     else
00397     {
00398         return it->second;
00399     }
00400 }
```

行番号	処理内容

359	ファイルから読み込んだ情報は、PropProfile 構造体の中に時間でソートされ文字列で格納されている。
	vector<pathLossMatrixValue>　matrixList;
	struct pathLossMatrixValue{
	　clocktype simTime;
	　string values;
	};
	
	一方、使用するパスロスマトリックスの情報は PartitionData 構造体に格納されている。
	map <pair<NodeId, NodeId>, double>** pathLossMatrix;
	int plCurrentIndex;
	clocktype plNextLoadTime;
	つまり、使用する時間のデータだけを map で保持しており、次に map を更新する時間を plNextLoadTime に設定している。

367	シミュレーション時間が map 更新時間に達した場合に、その時間で採用するパスロスマトリックスデータを更新する。
375	周波数、送受信ノードの組合せでパスロス値を検索する。
381	存在しない場合は、逆方向で検索する。 つまり、パスロスマトリックスファイルでの設定は、方向性を持って設定するのが良いが方向性が無い場合は、一方向のみの設定でもよい。
389	データが無い場合は HIGH_PATHLOSS=1000dB となる。

3.5.1.5 Okumura-Hata Propagation Model

観測データ(観測地：東京)に基づく曲線近似により、パスロス値を算出する macro-cellular system で有用なモデル。

経路損失は、以下の式で算出される

$$\text{Pathloss} = A + B * \log10(d) + C$$

A,B,C：地形データにより決定される値
d：送信端末と受信端末間の距離

QualNet では、PROPAGATION-OKUMURA-HATA-ENVIROMENT というサブパラメータで以下の環境の選択肢が用意されており、各環境に応じた係数がソースコードに記述されている。
QualNet での設定可能な環境は以下となる。
“OPEN_RURAL” “QUASI_OPEN_RURAL” “SUBURBAN” “URBAN” “METROPOLITAN”

本モデルは、以下の条件下で用いるのが望ましい。
・周波数は、150-1000MHz
・基地局のアンテナ高は、30-200m
・移動端末のアンテナ高は、1-10m
・基地局と端末間の距離は、1-20km

パスロス計算の実装箇所を以下に示す。

prop_hata.cpp

```
00098 double PathlossHata(double distance,
```

```
00099                       double waveLength,
00100                       float txAntennaHeight,
00101                       float rxAntennaHeight,
00102                       PropProfile *propProfile)
00103 {
00104
…
00148
00149         k = 0.0;
00150         a = (1.1 * log10(frequencyMhz) - 0.7) * h2
00151             - (1.56 * log10(frequencyMhz) - 0.8);
00152
00153         if (propProfile->propagationEnvironment == OPEN_RURAL) {
00154
00155             k = 4.78 * pow (log10(frequencyMhz),2.0)
00156                 - 18.33 * log10(frequencyMhz) + 40.94 ;
00157         }
00158         else if (propProfile->propagationEnvironment == QUASI_OPEN_RURAL) {
00159
00160             k = 4.78 * pow (log10(frequencyMhz),2.0)
00161                 - 18.33 * log10(frequencyMhz) + 35.94 ;
00162         }
00163         else if (propProfile->propagationEnvironment == SUBURBAN) {
00164
00165             k = 2.0 * pow (log10(frequencyMhz/28.0),2.0) + 5.4;
00166         }
00167         else if (propProfile->propagationEnvironment == METROPOLITAN){
00168             // building heights greater than 15 m
00169
00170             if (frequencyMhz >= 400.0){
00171
00172                 a = 3.2 * pow(log10(11.75 * h2),2.0) - 4.97;
00173             }
00174             else {
00175
00176                 a =8.29 * pow(log10(1.54 * h2),2.0) - 1.1;
00177             }
00178         }
00179
00180         pathloss_dB =
00181             69.55 + 26.16 * log10(frequencyMhz) - 13.83 * log10(h1) -
00182             a + (44.9 -6.55 * log10(h1)) * log10(distanceKm) - k;
00183     }
00184
00185     return pathloss_dB;
00186 }
```

行番号	処理内容
149-151	URBAN モデルを基準に、各エリア属性で補正(a, k)を行う。

3.5.1.6 COST 231-Hata Propagation Model

Okumura-Hata Propagation Model を 1500-2000MHz へ拡張し実験に基づいたモデルである。都市エリアや郊外エリアに適用できる屋外伝搬モデルで、平らな地形で有効である。基地局のアンテナ高が周辺の建物よりも十分に高い場合に適用される。

特定の地形データを考慮したものではないため、COST 231-Walfish-Ikegami (COST-WI) Propagation Model モデルより精度が劣る。

本モデルは、以下の条件下で用いるのが望ましい。

・環境は、都市エリア(urban)、郊外エリア(suburban)
・周波数は、150-2000MHz

・基地局のアンテナ高は、30-200m
・移動端末のアンテナ高は、1-10m
・基地局と端末との距離は、1-20km

QualNet では、PROPAGATION-COST231-HATA-ENVIROMENT というサブパラメータで以下の環境の選択肢が用意されており、各環境に応じた係数がソースコードに記述されている。
QualNet での設定可能な環境は以下となる。
“SUBURBAN” “URBAN”

パスロス計算の実装箇所を以下に示す。

prop_cost_hata.cpp

```
00087 double PathlossCOST231Hata(double distance,
00088                            double waveLength,
00089                            float txAntennaHeight,
00090                            float rxAntennaHeight,
00091                            PropProfile *propProfile)
00092 {
00093
…
00123         if (!((frequencyMhz > 1500.0) && (frequencyMhz < 2000.0)))
00124         {
00125             sprintf(errorStr, "Frequency = %f; Not in recommended"
00126                     "range [1500:2000]M.Hz\n", frequencyMhz);
00127             ERROR_ReportWarning(errorStr);
00128         }
00129
00130         a = (1.1 * log10(frequencyMhz) - 0.7) * h2
00131             - (1.56 * log10(frequencyMhz) - 0.8);
00132
00133         if (propProfile->propagationEnvironment == SUBURBAN) {
00134
00135             k = 0.0 ;
00136         }
00137         else if (propProfile->propagationEnvironment == URBAN) {
00138
00139             k = 3.0;
00140         }
00141
00142         pathloss_dB = 46.33 + 33.9 * log10(frequencyMhz) - 13.82 * log10(h1)
-
00143                       a + (44.9 - 6.55 * log10(h1)) * log10(distanceKm) + k;
00144
00145     }
00146
00147     return pathloss_dB;
00148 }
```

3.5.1.7 COST 231-Walfish-Ikegami (COST-WI) Propagation Model

回折理論を用いて建物高や道路幅等の市街地の状況を考慮したモデルである。基地局アンテナが建物の屋上にあるような場合に、高い精度が期待できるモデル

経路損失は、以下の式で算出される。

$$P_{PATHLOSS} = P_0 + P_{rts} + P_{msd}$$

$$P_0 = 32.4 + 20 * \log_{10}D + 20 * \log_{10}f$$
$$P_{rts} = -16.9 - 10 * \log_{10}w + 10 * \log_{10}f + 20 * \log_{10}\Delta h_m$$

$$+\begin{cases} -10+0.354\theta & (0 \le \theta < 35°) \\ 2.5+0.075(\theta-35) & (35° \le \theta < 55°) \\ 4.0-0.114(\theta-55) & (55° \le \theta < 90°) \end{cases}$$

$$P_{rts} = 54 - 18 * \log_{10}(1+\Delta h_b) + 18 * \log_{10} D + 9 * \log_{10} b$$
$$+\begin{cases} \left[-4+0.7*\left(\frac{f}{925}-1\right)\right] * \log_{10} f & \text{(urban)} \\ \left[-4+1.5*\left(\frac{f}{925}-1\right)\right] * \log_{10} f & \text{(metropolitan)} \end{cases}$$

$\Delta h_b = h_b - h_{roof} \quad (h_b > h_{roof})$
$\Delta h_m = h_{roof} - h_m \quad (h_{roof} > h_m)$

D：送受信点間距離[m]
f：周波数[MHz]
b：建物間隔[m]
w：道路幅[m]
θ：道路角[度]
h_{roof}：建物高[m]
h_b：基地局アンテナ高[m]
h_m：移動局アンテナ高[m]

本モデルは、以下の条件下で用いるのが望ましい。

・環境は、都市エリア(urban)、大都市エリア(metropolitan)
・周波数は、800-2000MHz
・基地局のアンテナ高は、4-70m
・移動端末のアンテナ高は、1-3m
・基地局と端末との距離は、1-5km

QualNet では、PROPAGATION-COST231-WALFISH-IKEGAMI-ENVIROMENT というサブパラメータで以下の環境の選択肢が用意されており、各環境に応じた係数がソースコードに記述されている。

QualNet での設定可能な環境は以下となる。

“URBAN” “METROPOLITAN”

COST-WI モデルに関連するソースコードは、\ libraries\urban\src\ prop_cost_wi.cpp

prop_cost_wi.cpp

```
00168 double PathlossCOST231_WI(Node *node,
00169                           double distance,
00170                           double waveLength,
00171                           float txAntennaHeight,
00172                           float rxAntennaHeight,
00173                           PropProfile *propProfile)
00174 {
…
00206 // Orientation always defaulted to 90 degrees: Ref: Principles of Mobile
00207 // Communication by G.L.Stuber
00208
00209     double  orientation = 90.0;
00210     //UrbanProp_SetOrientation(propProfile, Orientation);
00211
…
00232
00233         pathloss_dB = COST231_WI_NLoS (NumOfBuildingsInPath,
```

```
00234                                           distanceKm,
00235                                           frequencyMhz,
00236                                           h1,
00237                                           h2,
00238                                           orientation,
00239                                           aveRoofHeight,
00240                                           streetWidth,
00241                                           buildingSeparation,
00242                                           environment);
…
00248
00249     return pathloss_dB;
00250 }
```

行番号	処理内容
206-211	道路角(基地局～移動局方向と道路方向との成す角)は固定値で 90 度としている。
233	実際の計算は COST231_WI_NLoS 関数で算出。

```
00389 double COST231_WI_NLoS (int NumOfBuildingsInPath,
00390                         double distanceKm,
00391                         double frequencyMhz,
00392                         double h1,
00393                         double h2,
00394                         double orientation,
00395                         double roofHeight,
00396                         double streetWidth,
00397                         double buildingSeparation,
00398                         PropagationEnvironment environment)
00399 {
…
00405     if (NumOfBuildingsInPath < 1) {
00406
00407         pathloss_dB = COST231_WI_LoS(distanceKm, frequencyMhz);
00408     }
00409     else {
00410
…
00420
00421         streetDiffractionLoss = COST231_WI_RooftoStreetLoss(
00422                                     frequencyMhz,
00423                                     orientation,
00424                                     h2,
00425                                     roofHeight,
00426                                     streetWidth);
00427
…
00433         multiscreenDiffractionLoss = COST231_WI_MultiScreenLoss(
00434                                         distanceKm,
00435                                         frequencyMhz,
00436                                         h1,
00437                                         roofHeight,
00438                                         buildingSeparation,
00439                                         environment);
00440
00441         freeSpaceLoss =
00442             32.4 + 20.0 * log10(distanceKm) + 20.0 * log10(frequencyMhz);
00443
00444         if ((streetDiffractionLoss + multiscreenDiffractionLoss) > 0.0) {
00445
00446             pathloss_dB = freeSpaceLoss + streetDiffractionLoss
00447                           + multiscreenDiffractionLoss;
00448         }
00449         else {
00450             pathloss_dB = freeSpaceLoss;
00451         }
```

```
00452
…
00458     }
00459
00460     return pathloss_dB;
00461
00462 }
```

行番号	処理内容
405-410	見通し有りの場合
421-427	基地局と移動局間にある建物による回折損失(Rooftop to street propagation loss) を算出する。
433-439	移動局近傍の建物と移動局間の回折や反射に伴う損失(Multi screen diffraction loss)を算出する。
441	基地局と移動局間の自由空間損失を計算し、上記の損失を付加して伝搬損失を求める。

3.5.1.8 Automatic Model Selection

送受信端末の位置や urban エリアの地形特徴(つまり周辺ビル群の配置)を元に、自動的に適切なパスロスモデルを選択する。以下の Street Microcell Propagation Model でもビル群の配置情報(urban terrain)を元に伝搬計算を行っている。urban terrain の作り方や設定方法は、QualNet-5.1-Wireless-ModelLibrary.pdf P.337 「8.5 Urban Terrain Data Format」、及び QualNet-5.1-UsersGuide.pdf P.370 「D.4 Urban Grid Script」を参照のこと。

判定するのは以下の状態である。
IsLoS …送受信点間の見通し状況
IsCanyon …送受信端末のアンテナ高が周辺のビルより十分低いか(峡谷の底にいるのか)

送受信端末の状況により採用されるパスロスモデルを、以下の表 3-14　Automatic Model の判別表 3-14 に示す。

表 3-14　Automatic Model の判別

No.	IsLoS	Node1 IsCanyon	Node2 IsCanyon	Path loss Model
1	TRUE	TRUE	TRUE	Street Microcell(LoS)
2	TRUE	TRUE	FALSE	COST Walfish-Ikegami(LoS)
3	TRUE	FALSE	TRUE	COST Walfish-Ikegami(LoS)
4	TRUE	FALSE	FALSE	Free-space
5	FALSE	TRUE	TRUE	Street Mobile-to-mobile or Street Microcell (NLoS)
6	FALSE	TRUE	FALSE	COST Walfish-Ikegami(NLoS)
7	FALSE	FALSE	TRUE	COST Walfish-Ikegami(NLoS)
8	FALSE	FALSE	FALSE	このケースは Urban では考えられない

なお、マニュアルには記載が無いが、ソースコードを解読すると、まず最初に屋内か否かをチェックしているのがわかる。つまり、Urban Terrain で設定したビル内部に送受信端末が位置している場合は、屋内空間では ITU_R 及び COST231 の屋内伝搬推定式に沿ってパスロスを計算しパスが屋外に出ると上記の表に従いパスロス計算を行っている。この部分は非常に専門的であり、また引用元も明確でないので、本書では詳細は解説しない。

3.5.1.9 Street Microcell Propagation Model

urban canyon、つまりビルの間に面した通りにおける送信端末と受信端末間の経路損失を算出するモデル。

サブパラメータは以下の通り。

PROPAGATION-STREET-MICROCELL-ENVIRONMENT …見通し状況("NLOS", "LOS")

本モデルの詳細は、以下の文献を参考に実装されている。
Gordon L. Stüber, "Principles of Mobile Communication", Second Edition, 2002
P.Harley, "Short distance attenuation measurements at 900 MHz and 1.8 gHz using low antenna heights for microcells", IEEE JSAC, vol.7, pp.5-11, Jan 1989.

3.5.1.9.1 Street Mobile-to-mobile Propagation Model

送信端末と受信端末が共に urban canyon の中、つまり複数ビルの谷間の路上に配置されている場合を想定した伝搬モデルである。

サブパラメータは以下の通り。
NUM-OF-BUILDINGS-IN-PATH …送受信点間に存在すると仮定するビルの数
PROPAGATION-ROOF-HEIGHT …障害物となるビルの平均高さ
PROPAGATION-STREET-WIDTH …道路の平均幅

本モデルの詳細は、以下の文献を参考に実装されている。
Gordon L. Stüber, "Principles of Mobile Communication", Second Edition, 2002

3.5.1.10 Suburban Propagation Model

このモデルでは、Suburban 環境下に特化した伝搬モデルである。地形や木の葉の影響を考慮している。

設定するサブパラメータは以下の通りの 2 種類である。
PROPAGATION-TERRAIN-TYPE…地形のタイプ("FLAT" or "HILLY")
PROPAGATION-PERCENT-AREA-COVERED-BY-VEGETATION…植物のカバー率

しかし実際の計算では以下の 3 つのタイプに変換して条件分岐している。
HILLY_TERRAIN_WITH_MOD_TO_HEAVY_TREE_DENSITY
　　地形のタイプ="HILLY"
FLAT_TERRAIN_WITH_MOD_TO_HEAVY_TREE_DENSITY
　　地形のタイプ=" FLAT"、かつ植物のカバー率≧DEFAULT_VEG_CUTOFF_PERCENT
FLAT_TERRAIN_WITH_LIGHT_TREE_DENSITY
　　地形のタイプ=" FLAT"、かつ植物のカバー率＜DEFAULT_VEG_CUTOFF_PERCENT
ここで、DEFAULT_VEG_CUTOFF_PERCENT=65 と固定値で設定されている。

Suburban モデルに関連するソースコードは、

$QUALNET_HOME/libraries/urban/src/prop_suburban.cpp にある。

現段階では引用元の計算式が不明であるが、ソースコードを参照してわかる通り、上記のタイプ別に係数が決められてパスロスを計算している。

3.5.2 Shadowing

3.5.2.1 設定値

PROPAGATION-SHADOWING-MODEL での設定値は以下の通りである。

表 3-15 PROPAGATION-SHADOWING-MODEL

モデル名	設定値
None	NONE
Constant	CONSTANT
Lognormal	LOGNORMAL

それぞれの詳細は以下の通り。

- **None**
  Shadowing による減衰は無い(= 0dB)。
- **Constant**
  常に一定の減衰が生じるモデルである。*PROPAGATION-SHADOWING-MEAN* で設定した値[dB]が減衰量となる。なお、None が設定されている場合は、Constant が設定されておりかつ *PROPAGATION-SHADOWING-MEAN* が 0 に設定されている場合と全く同じ処理となる。
- **Lognormal**
  PROPAGATION-SHADOWING-MEAN で設定した値[dB]を平均とする対数正規分布によって減衰量が決まる。

3.5.2.2 初期化処理

上記設定項目は以下の通り、PROP 層の初期化時に反映される。

propagation.cpp

```
2802 void PROP_Init(Node *node, int channelIndex, NodeInput *nodeInput) {

.... 中略 ....

2814     propData->shadowingDistribution.setSeed(
2815         node->globalSeed,
2816         node->nodeId,
2817         channelIndex);
2818     if (propProfile->shadowingModel == CONSTANT) {
2819         propData->shadowingDistribution.setDistributionDeterministic(
2820             propProfile->shadowingMean_dB);
2821     }
2822     else { // propProfile->shadowingModel == LOGNORMAL
2823         propData->shadowingDistribution.setDistributionGaussian(
2824             propProfile->shadowingMean_dB);
2825     }
```

ここで、shadowingDistribution は RandomDistribution テンプレートクラスのインスタンス(double 型)であり、QualNet に於ける汎用乱数生成器である。

※setSeed ではシードとして自身の nodeId を与えており、この乱数生成器は受信機側となった場合に使用される。QualNet ではマルチスレッド実行した場合、例えば異なるスレッド上で処理される TxA と TxB からの電波を、あるノードがほぼ同時刻に受信するようなケースにおいて、それらの伝搬計算順序が入れ替わる可能性がある。計算順序の入れ替わりによる論理的な整合性は考慮されているが、shadowing の計算に関しては上記のような受信側ノード ID のみ使用するシードの与え方が原因で、乱数の取得順序が入れ替わり、実行のたびに計算結果が変わるケースがあることに注意されたい。

random.h

```
139 // /**
140 // CLASS        :: RandomDistribution
141 // DESCRIPTION  :: a template class for generating various types of random
142 //                 distributions with different return types.  Primarily
143 //                 used by models that allow the user to specify the type
144 //                 of distribution as a configuration option.
145 // **/
146 template <class T>
147 class RandomDistribution
```

メンバ関数 setDistributionDeterministic()は毎回同じ値(shadowingMean_dB)を生成するように乱数生成器をセットし、setDistributionGaussian()は正規分布(平均値= shadowingMean_dB)による乱数を生成するように乱数生成器をセットする。

※このモデルでは単純な正規分布であり、例えば空間的な相関(ある座標の十分近傍では shadowing の値も近くなる、など)を考慮したような作りとはなっていない点に注意されたい。

3.5.2.3 計算処理

shadowing の計算は pathloss の計算時にあわせて行われる。

propagation.cpp

```
218 void PROP_CalculatePathloss(
 219     Node* node,
 220     NodeId txNodeId,
 221     NodeId rxNodeId,
 222     int channelIndex,
 223     double wavelength,
 224     float txAntennaHeight,
 225     float rxAntennaHeight,
 226     PropPathProfile *pathProfile,
 227     double* pathloss_dB,
 228     bool forBinning)
 229 {

.... 中略 ....

 323              shadowing_dB = propData->shadowingDistribution.getRandomNumber();

.... 中略 ....

 344             *pathloss_dB += shadowing_dB;
```

前述の RandomDistribution インスタンスから getRandomNumber()を呼び出すことで、初期化時にセットしたポリシーに従い shadowing 値を取得し、最終的に pathloss の値に合算される。

なお前述の Pathloss と同様、Shadowing は送受信点の位置が変わらない場合は、最新の計算結果を再利用し再計算は行われない。

3.5.3 Fading

QualNet では、レイリーフェージング及びライスフェージングモデルをサポートしている。更に、レイリーフェージングモデルには、ドップラー周波数が一定のモデルと、ノードの移動速度に応じたドップラー周波数を適用可能なモデルの、2 つのモデルがある。

使用するフェージングモデルは、*PROPAGATION-FADING-MODEL* で設定を行う。サポートされているモデル及びパラメータの設定値は以下の通りである。

表 3-16 フェージングモデル及び設定値

モデル名	モデルの概要	設定値
None	フェージング無し	NONE
Rayleigh	レイリーフェージング	RAYLEIGH
Ricean	ライスフェージング	RICEAN
High Speed Fading	ノードの移動速度に応じたドップラー周波数のレイリーフェージング	FAST-RAYLEIGH

Rayleigh 及び Ricean モデルについては、*PROPAGATION-FADING-MAX-VELOCITY* パラメータで、移動速度[m/s]を指定することで、ドップラー周波数を指定する必要がある。(nodes ファイルで指定する、実際のノードの移動速度とは関係ない)

また、None 以外に設定する際には、*PROPAGATION-FADING-GAUSSIAN-COMPONENTS-FILE* にてファイルを指定する必要がある。

3.5.3.1 QualNet でのフェージングの模擬

QualNet ではフェージングの生成をプログラム内部で 0 から行うのではなく、予め外部で作成したフェージング変動を表す時系列データを基に生成を行う。外部から与えるファイルは、単一のドップラー周波数のレイリーフェージング変動である。実際シミュレーション中に適用される様々なドップラー周波数のフェージングやレイリーフェージング以外のフェージング変動への換算をシミュレータ内部で行っている。

フェージングの時系列データを含む外部ファイルは、QualNet では “Gaussian Components File” と呼んでいる。フェージング変動に用いる Gaussian Components File は、パラメータ *PROPAGATION-FADING-GAUSSIAN-COMPONENTS-FILE* で指定を行う。

3.5.3.1.1 Gaussian Components File

Gaussian Components File には、時系列データの情報を与える設定項目及び複素数値の時系列値が記載される。

設定項目は以下の通り。

表 3-17 Gaussian Components File の

設定項目	単位	説明
SAMPLING-RATE	Hz	時系列データのサンプリングレート
BASE-DOPPLER-FREQUENCY	Hz	ドップラー周波数[1]
NUMBER-OF-GAUSSIAN-COMPONENTS	-	時系列データ数

[1]シミュレーション中のフェージング変動に適用されるドップラー周波数ではなく、あくまで本ファイルでフェージングの種として与える時系列データのドップラー周波数を指定するもの。この意味で、”**BASE-**DOPPLER-FREQUENCY”という名称になっている。

以降、*NUMBER-OF-GAUSSIAN-COMPONENTS* で指定された数の複素数(実部・虚部)が記載される。(詳細は「QualNet-5.1-Wireless-ModelLibrary.pdf」の「Format of the Gaussian Components File」を参照のこと。)

QualNet には Rayleigh フェージングの時系列データを含む Gaussian Components File が 1 つ付属している。（scenario\default ディレクトリの下の “default.fading”[2])。

```
・・・略・・・

SAMPLING-RATE 1000
BASE-DOPPLER-FREQUENCY 30.0
NUMBER-OF-GAUSSIAN-COMPONENTS 16384

 -5.6482112e-001 -1.2675110e+000
 -5.7047958e-001 -1.0847877e+000
 -5.6146223e-001 -8.8065119e-001

・・・略・・・
```

特別な理由がない限り、フェージングを適用したい場合には、このファイルをそのまま用いれば十分であろう。

図 3-26 は、default.fading ファイルの実部、虚部の値をプロットしたグラフである。(500-1000sample 目)

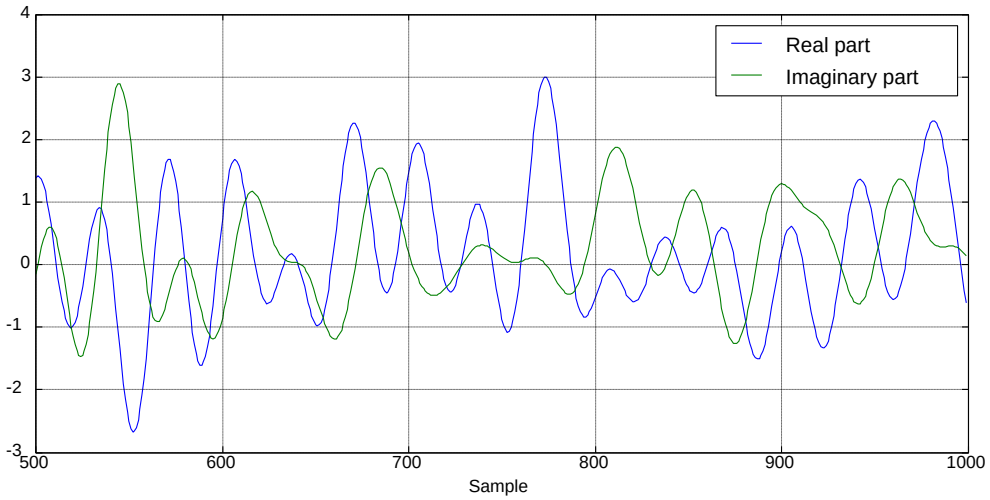


図 3-26 default.fading

default.fading は、*SAMPLING-RATE* が 1000Hz,*BASE-DOPPLER-FREQUENCY* が 30Hz である。したがって、このファイルは、30Hz のドップラー周波数の Rayleigh フェージングを 1ms 間隔でサンプリングした時系列のデータということになる。

[2]「このファイル内のデータは、”Theodore Rappaport, Wireless Communications: Principles and Practice, 2nd Edition”を参考に生成した」という旨の回答が、SNT 社のユーザフォーラムサイトにおいて 2008 年 10 月 11 日に公式の技術サポート担当からなされている。

図 3-27 は default.fading のフェージングゲインを表 3-18 の式に従って算出し横軸を時間としてプロットしたグラフである。

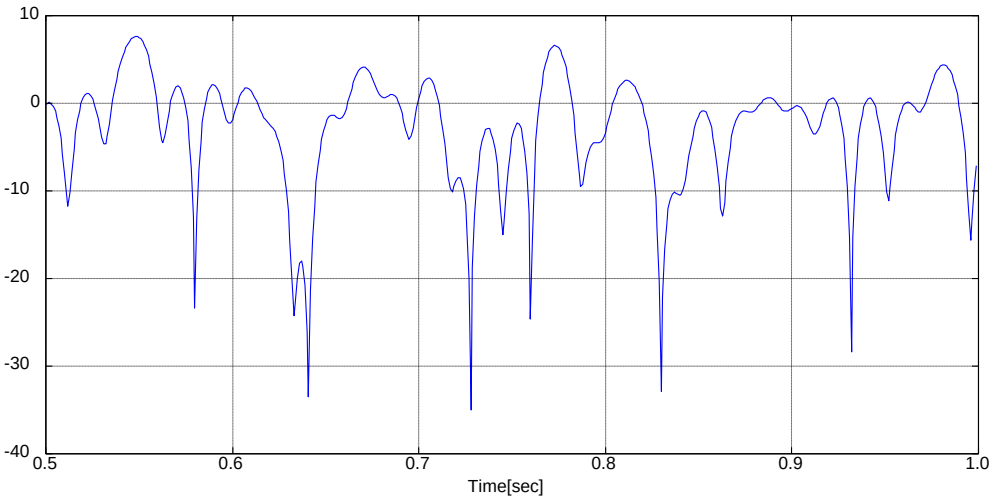


図 3-27 default.fading のフェージングゲインの様子

3.5.3.2 フェージングの計算方法

フェージングを計算している箇所は、PROP_CalculateFading 関数である。モデルの種類によらず、この関数内でフェージングの計算が行われる。

この関数は、PROP_ReleaseSignal で信号が送信されるたび、QualNet カーネルから呼び出されてフェージングの計算を行う。図 3-28 に示すように、全ての送受信ノードのペアに関して呼び出される。呼び出されるタイミングは、送信元ノードから信号の先頭が発射された瞬間である。

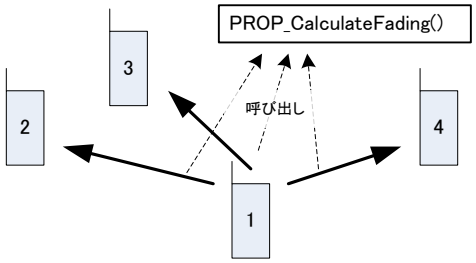


図 3-28 PROP_CalculateFading の呼び出し

各送受信ノード間のフェージングは、与えられた GaussianComponentFile の現在時刻に応じた位置(インデックス)の値を基に算出される。ただし、全ての送受信ノード間のフェージングが同じ変動となるのを避けるために、

1) 送信元ノード ID
2) 送信先ノード ID
3) チャネル番号

の 3 つの値の組から一意に定まるランダムな位置(Index) (これを Starting Point と呼ぶ)を時刻 0 に当たる場所とし、そこを起点として時刻が進むごとに Gaussian Component ファイルを ”なぞって” フェージングを計算するようになっている。このようにすることで、任意の 2 つの送受信ノード及びチャネル番号 で表される伝搬路の、疑似的にバラバラに変動するフェージングを模擬している。フェージングの計算イメージを図 3-7 に示す。なお、Gaussian ComponentFile の最後まで到達したら、再び先頭に戻るようになっている。

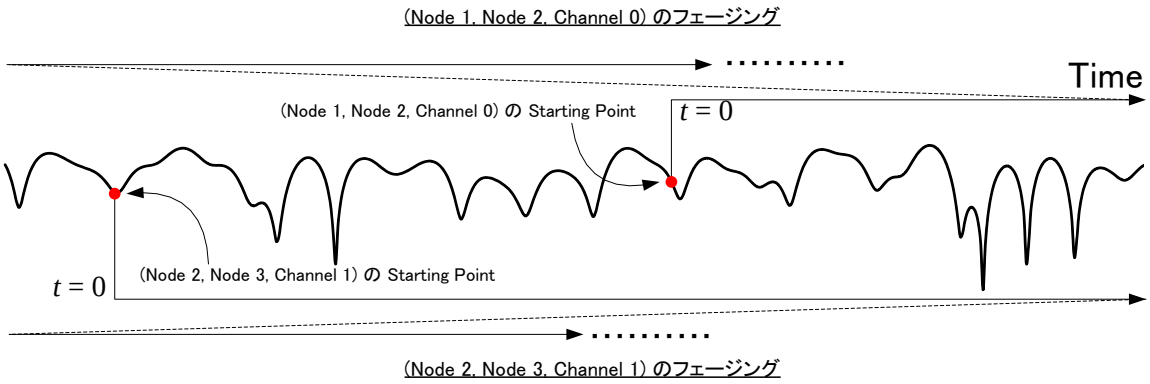


図 3-29 フェージング計算のイメージ

また、ドップラー周波数に依存する、フェージングの変動の速さの違いは、プログラム的には、Gaussian Component File を“なぞるスピード”を変えることで、実現を行っている。

最終的に、ある時刻の Gaussian Component File のインデックスは以下の式であらわされる。

$$StartingPoint = (送信ノードID, 受信ノードID, チャネル番号)\ で一意に決まる乱数$$
$$Index = mod(StartingPoint + (現在時刻[sec] * StretchingFactor), サンプル数)$$

StretchingFactor が”なぞるスピード”を表す変数である。Rayelgih 及び Ricean モデルの場合は、

$$StretchingFactor = SamplingRate[Hz] \times \frac{DopplerFrequency[Hz]}{BaseDopplerFrequency[Hz]}$$

で計算される。High Speed Fading モデルの場合には、その時刻における送受信ノードの移動速度から計算される。

上記で計算された、Index の位置の Gaussian Component File の値を用いて、フェージングの値を計算する。今、Gaussian Component File の Index 番目の要素の実数部、虚数部をそれぞれ R, I と表すと、各モデルのフェージングゲインは以下の式であらわされる。

表 3-18 各モデルのフェージングゲインの計算式

モデル	フェージングゲイン[dB]
None	0
Rayleigh	$10\log_{10}\left(\frac{R^2+I^2}{2}\right)$
Ricean	$10\log_{10}\left(\frac{\left(R+\sqrt{2K}\right)^2+I^2}{2(K+1)}\right)$ K : K ファクタ. パラメータで指定.
High Speed Fading	$10\log_{10}\left(\frac{R^2+I^2}{2}\right)$

3.5.3.3 PROP_CalculateFading 関数の解説

以降では、PROP_CalculateFading 関数のソースコードを追いかけながらフェージング計算の詳細を説明する。

フェージングモデルの種類は、PropProfile 構造体の fadingModel メンバ変数に格納されている。1222 行目から始まる if-else 文は、フェージングモデル別の計算処理に分岐させるための条件文である。よく見るとこの if-else 文では、「RICEAN かどうか？」しか判定をしておらず、RAYLEIGH かどうかを判定する文がないことがわかる。実は、プログラムの内部的には、フェージングモデルは、全てライスフェージングとして扱われる。なぜならば、レイリーフェージングはライスフェージングの特別なケース(直接波なし)だからである。

propagation.cpp

```
1206 // assuming here that the receiving node (node 2) is always local,
      while transmitter might be remote.
1207 // also assuming that fading stretching factor is the same for both nodes
1208 void PROP_CalculateFading(
1209     Message* signalMsg,
1210     PropTxInfo* propTxInfo,
1211     Node* node2,
1212     int channelIndex,
1213     clocktype currentTime,
1214     float* fading_dB,
1215     double* channelReal,
1216     double* channelImag)
1217 {
1218     PropChannel* propChannel = node2->partitionData->propChannel;
1219     PropProfile* propProfile = propChannel[channelIndex].profile;
1220     PropProfile* propProfile0 = propChannel[0].profile;
1221
1222     if (propProfile->fadingModel == RICEAN) {
1223         int arrayIndex;
1224         double arrayIndexInDouble;
1225         double value1, value2;
1226
1227         const float kFactor = (float)propProfile->kFactor;
1228         const int numGaussianComponents = propProfile0->numGaussianComponents;
1229         const int startingPoint =
1230             RandomizeGaussianComponentStartingPoint(
1231                 propTxInfo->txNodeId, node2->nodeId, channelIndex,
1232                 numGaussianComponents);
1233
```

1229 行目で RandomizeGaussianComponentStartingPoint() を呼んでいる箇所が、送信ノード ID, 受信ノード ID, チャネル番号に対して、一意のランダムな Starting Point を計算している箇所である。

RandomizeGaussianComponentStartingPoint 関数は以下のような内容になっている。nodeId1, nodeId2 が、送受信ノードの ID である。1197, 1198 行目で seed を取得する関数、RANDOM_SetSeed の引数に Node ID を渡す際、MIN, MAX を取っている。これは、チャネルが対称(Node 1 から Node 2 へ送った信号と、Node 2 から Node 1 へ送った信号に作用するフェージングは同じ)であることを考慮している。

propagation.cpp

```
1187 static
1188 int RandomizeGaussianComponentStartingPoint(
1189     NodeAddress nodeId1,
1190     NodeAddress nodeId2,
1191     int         channelIndex,
1192     int         arraySize)
```

```
1193 {
1194     RandomSeed seed;
1195
1196     RANDOM_SetSeed(seed,
1197                    MIN(nodeId1,nodeId2),
1198                    MAX(nodeId1,nodeId2),
1199                    channelIndex);
1200
1201     return RANDOM_nrand(seed) % arraySize;
1202 }
```

1234 行目は、propProfile->motionEffectsEnabled という変数を見ている。High speed fading モデル(FAST-RAYLEIGH)を用いた場合、propProfile->motionEffectsEnabled は TRUE となっている。
このブロックで呼ばれる PROP_MotionObtainfadingStretchingFactor 関数は、非公開の関数であるが、内部では、そのブロックの下の 1242 行目で参照している fadingStretchingFactor の値を現在のノードの移動速度に応じた値に更新している。

1241 行目で計算するインデックスが、Starting Point からのインデックスのオフセットである。ここで、fadingStretchingFactor という値が、なぞるスピードを表す変数である。fadingStretchingFactor は、Rayleigh, Ricean モデルの場合は、PROP_Init 関数で計算が行われる。*PROPAGATION-FADING-MAX-VELOCITY* で設定された速度(ドップラー周波数)が大きければ、fadingStrechingFactor の値が大きくなり、なぞるスピードが速くなる。速度が 0 に設定された場合 stretchingFactor は 0 となり、結果なぞするスピードは 0 となる。この場合、 時間によらず常に同じフェージング値(Starting Point の位置のフェージング値)となる。

1245 行目は、少々わかりにくいが、単に numGaussianCOmponents で余剰を取っているだけである。
1249 行目で参照すべき GaussianComponentFile のインデックスを算出している。インデックスは Starting Point と時刻に応じたオフセット値の和となる。

1257 行目で、フェージングのゲイン[dB]を計算している。なお、Ricean フェージングにおける K ファクタ = 0 の場合が、レイリーフェージングであるため、1 つの式でまとめて計算している。

propagation.cpp

```
1233
1234         if (propProfile->motionEffectsEnabled){
1235
1236             PROP_MotionObtainfadingStretchingFactor(propTxInfo,
1237                                                     node2,
1238                                                     channelIndex);
1239         }
1240
1241         arrayIndexInDouble =
1242             node2->propData[channelIndex].fadingStretchingFactor *
1243             (double)currentTime;
1244
1245         arrayIndexInDouble -=
1246             (double)numGaussianComponents *
1247             floor(arrayIndexInDouble / (double)numGaussianComponents);
1248
1249         arrayIndex =
1250             (RoundToInt(arrayIndexInDouble) + startingPoint) %
1251             numGaussianComponents;
1252
1253         value1 = propProfile0->gaussianComponent1[arrayIndex] +
1254             sqrt(2.0 * kFactor);
1255         value2 = propProfile0->gaussianComponent2[arrayIndex];
1256
1257         *fading_dB =
```

```
1258             (float)IN_DB((value1 * value1 + value2 * value2) / (2.0 * (kFactor
+ 1)));
1259     }
1260     else {
1261         *fading_dB = 0.0;
1262     }
1263 }
```

コラム : High Speed Fading モデルの動作

図 3-30 は、High Speed Fading を用いた場合の、StretchingFactor と、フェージングゲインの様子を表したものである。ノードは片方は固定、もう片方は以下の速度でまっすぐ遠ざかっている場合の結果である。

- 時刻 0[sec] ～ 2[sec] : 1 m/sec
- 時刻 2[sec] ～ 4[sec] : 2 m/sec
- 時刻 4[sec] ～ 6[sec] : 0.5 m/sec

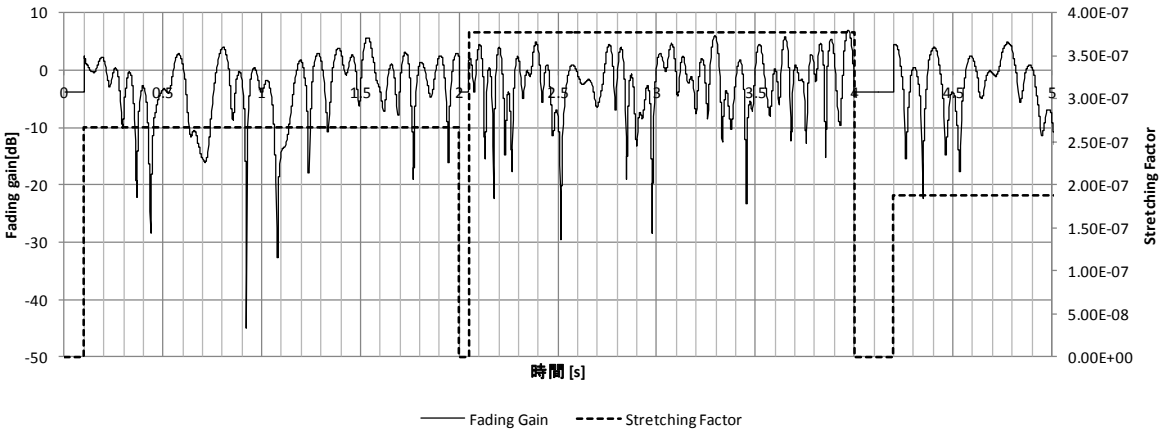


図 3-31 High Speed Fading モデルを用いた場合のフェージングの様子

グラフから、速度に応じて StretchingFactor の値が上下し、それに伴い、フェージングの変動速度が変化していることがわかる。

ただし、速度が変化する際に一瞬、StretchingFactor が 0 になりフェージングが止まる現象がみられる。StretchingFactor が 0 になる区間長は移動速度により異なることがわかる。QualNet の移動モデルには、”移動とみなす最低距離(分解能)”という概念がある。この値は、MOBILITY-POSITION-GRANULARITY というパラメータで指定することができる。上記の例では、MOBILITY-POSITION-GRANULARITY は 0.1[m] という値を用いた。StrechingFactor が 0 となっている期間は、実はその時の速度で MOBILITY-POSITION-GRANULARITY [m]だけ進むのに必要な時間と等しい。モビリティモデルの制約上、移動速度が変化した場合 MOBILITY-POSITION-GRANULARITY で指定した距離進むまでは、移動速度が 0 とみなされるようになっている。そのため、フェージングもストップするようになってしまっている。現状ではこの動作は、仕様上の制限事項となっている。