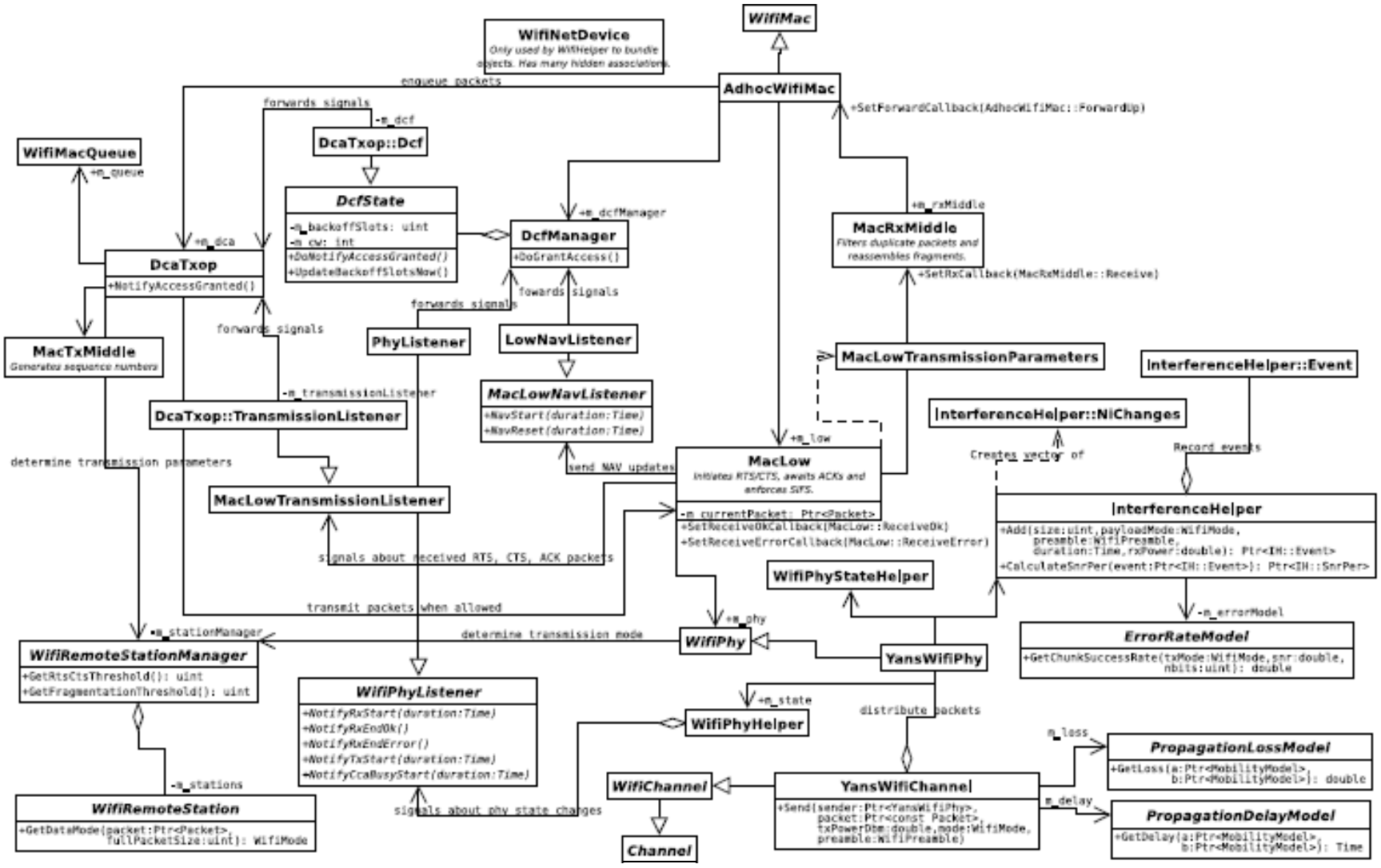
wifiモジュール

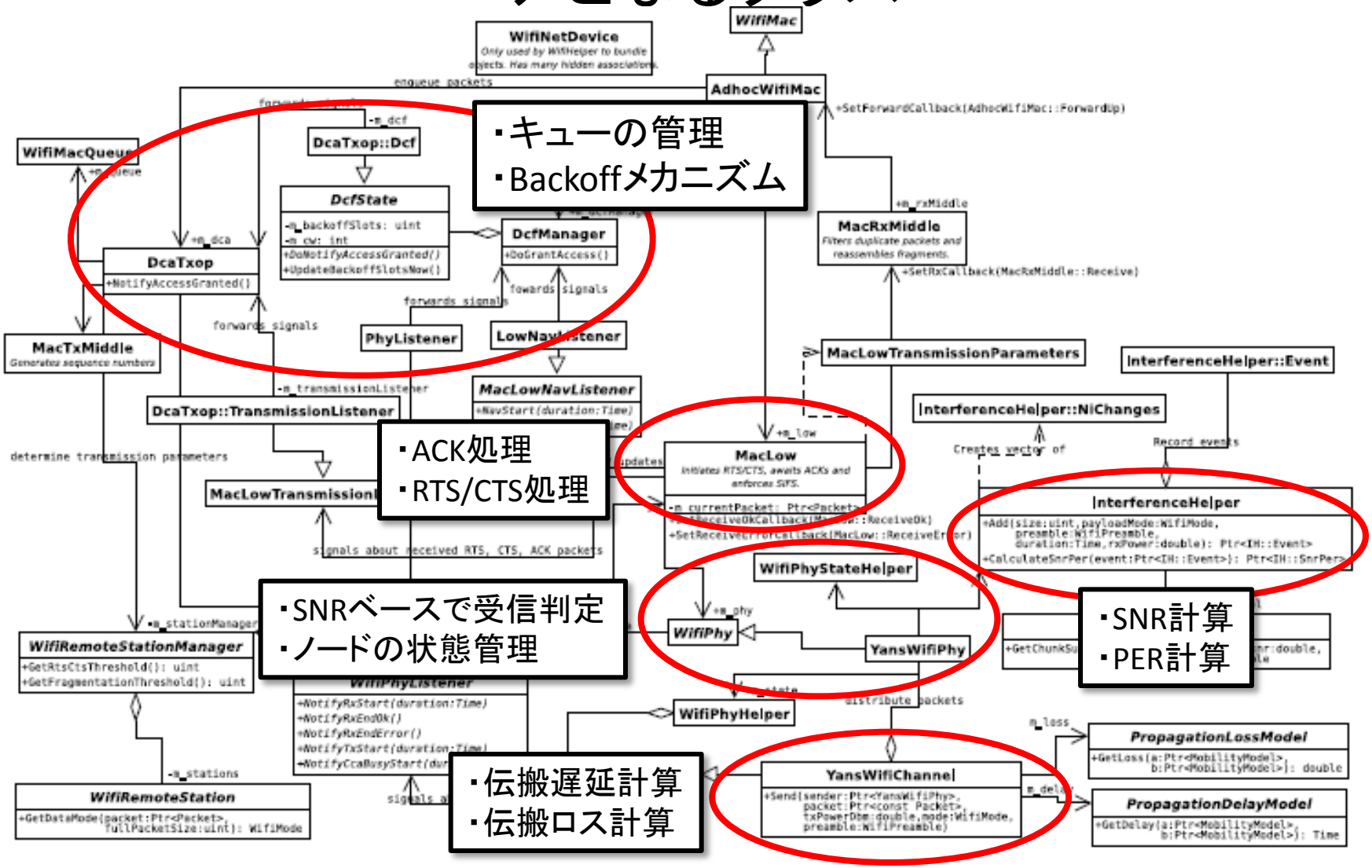
静岡大学 情報学研究科
杉山 佑介
2014/07/22

wifiモデルのクラス図



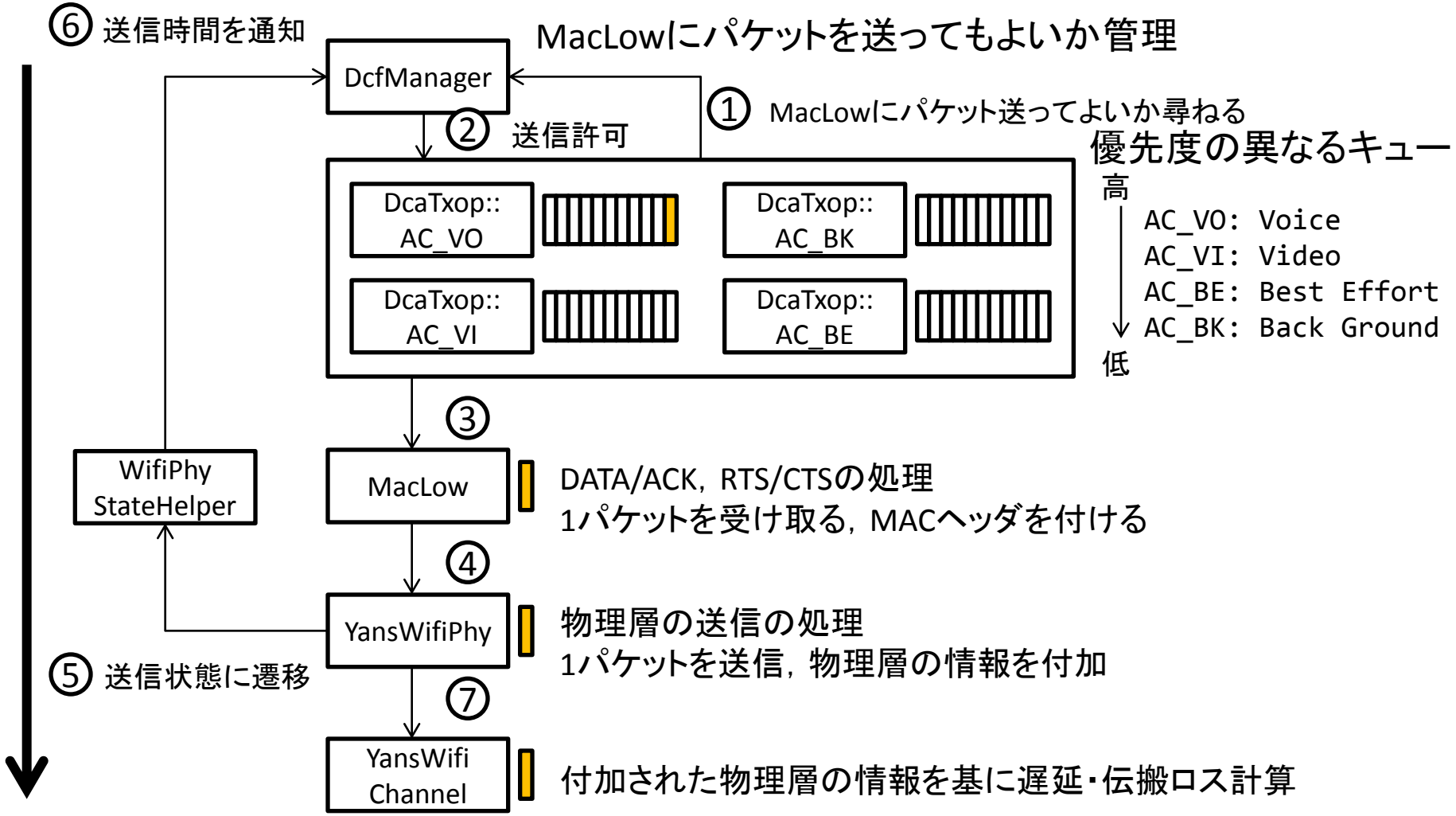
Timo B., Accuracy Enhancements of the 802.11 Model and EDCA QoS Extensions in ns-3

wifiモデルのクラス図
~コアとなるクラス~

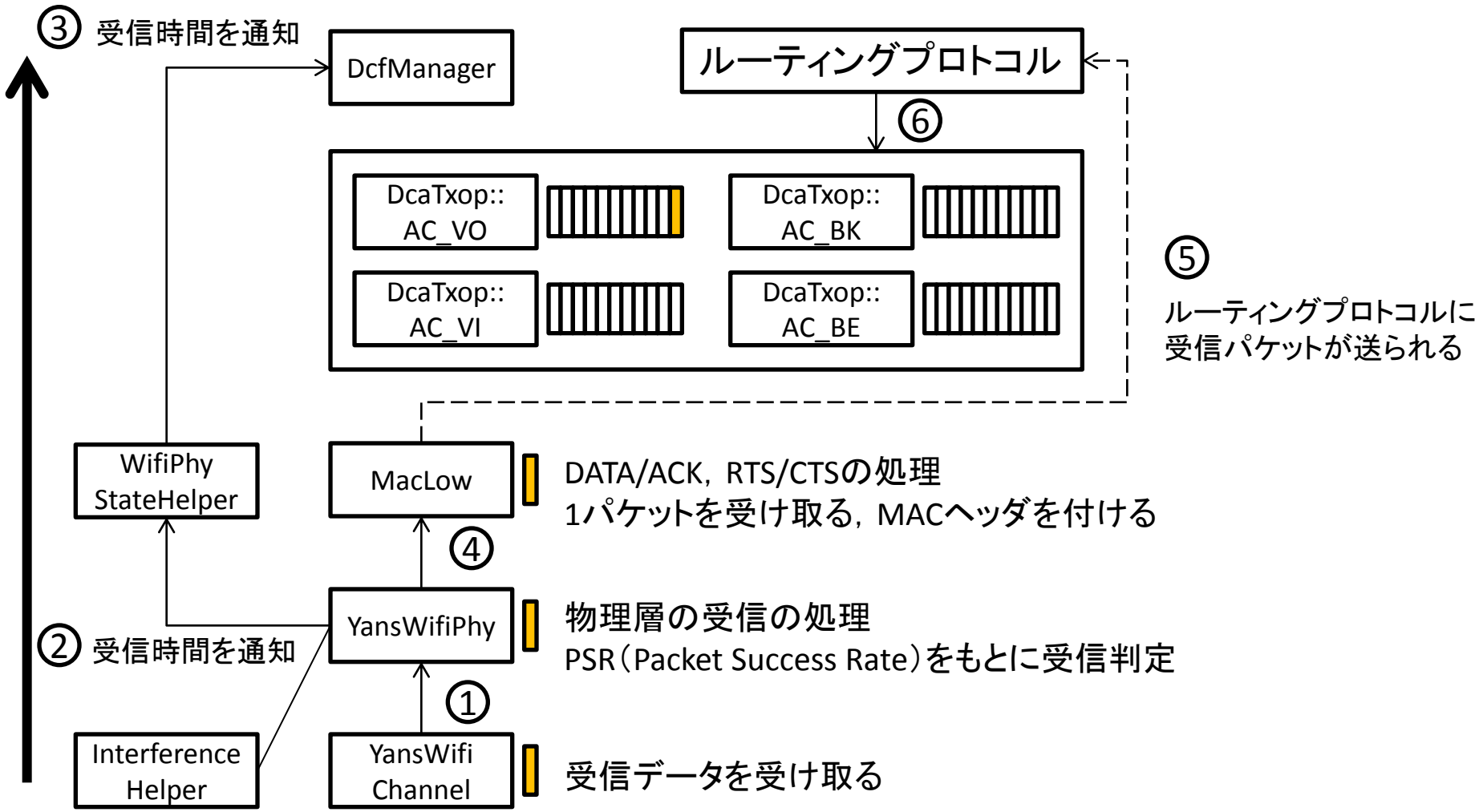


Timo B., Accuracy Enhancements of the 802.11 Model and EDCA QoS Extensions in ns-3

コアとなるクラスの関係 ~送信~

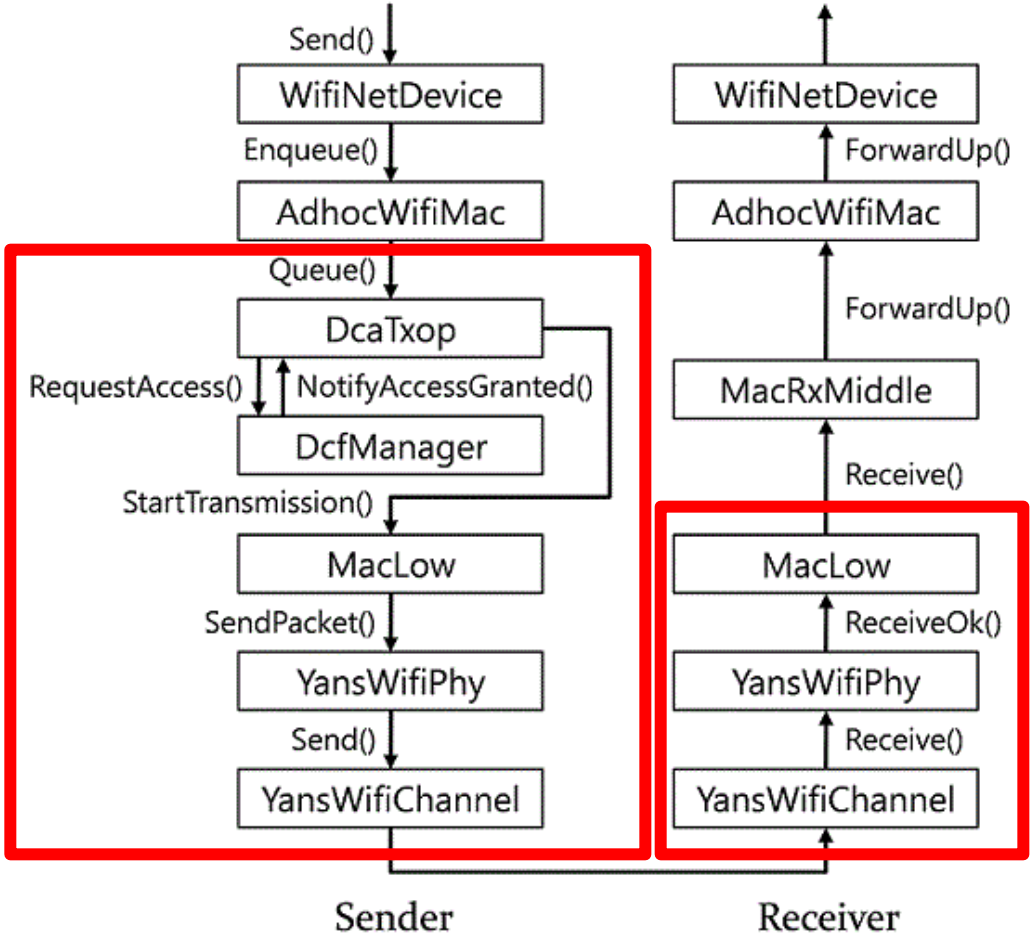


コアとなるクラスの関係 ~受信~



コアとなるクラス関係 ~関数レベル~

- キューにパケットが送られてからMAC層で処理するまでの流れ



http://www2.engr.arizona.edu/~junseok/ns3_wifi.htm

DATA送信からACK送信までの流れ
前置き

DcfManager: DcaTxopの管理（MacLowのアクセス権，MAC層の時間の管理）

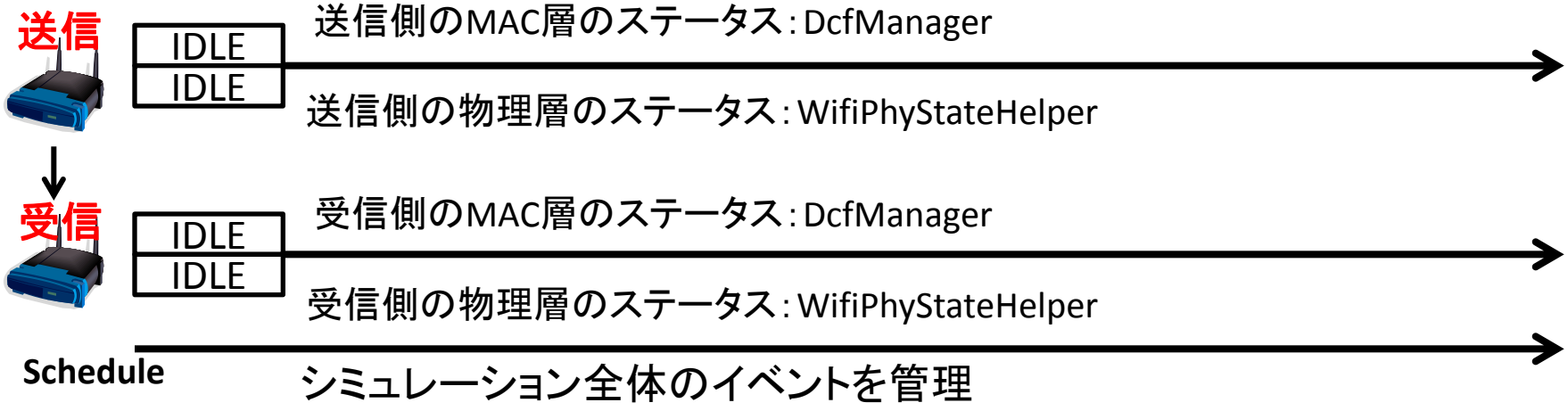
DcfTxop: 送信キュー，Backoffのメカニズム

MacLow: DATA/ACK，RTS/CTSの処理（再送など）

YansWifiPhy: SNRを基に受信判定

YansWifiChannel: 伝搬遅延，伝搬損失を計算して送信，各ノードへの接続

最も簡単なモデルであるIEEE 802.11aのQOS制御なしの場合を説明
1台の送信ノードから1台の受信ノードに送信



DcaTxop：送信

DcaTxop

Queue (packet, hdr)

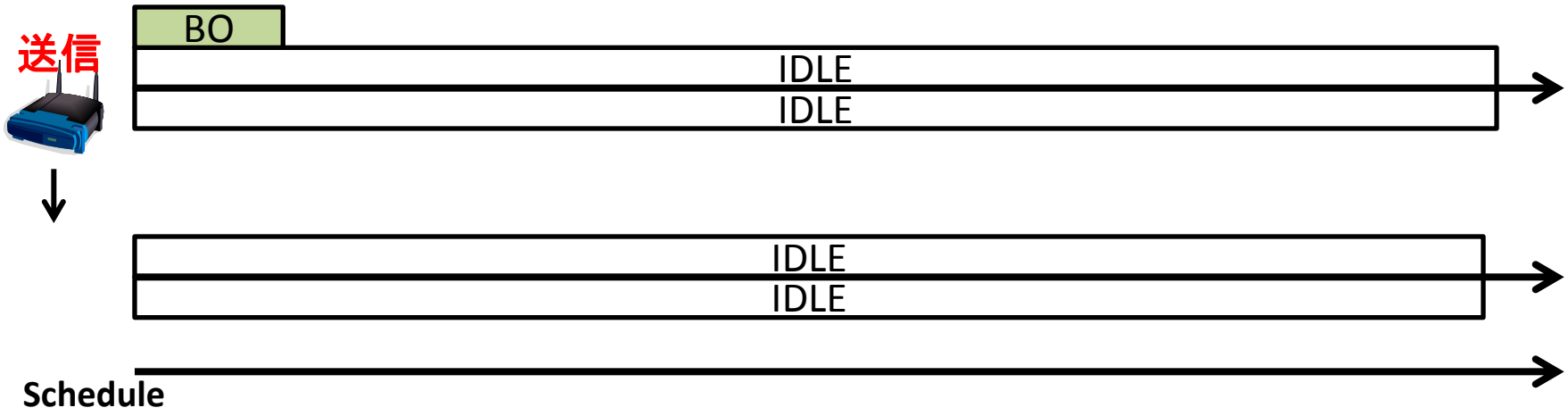
- 引数
  - packet: 送信パケット
  - hdr: 送信MACヘッダ
- 仕様：MAC層のキューにパケットを格納，DcfManagerにMacLowにアクセス要求を出す
  1. MAC層のキュー（m_queue）にpacketとhdrを格納
  2. StartAccessIfNeeded ()を呼び出す

DcaTxop：送信

DcaTxop

StartAccessIfNeeded ()

- 仕様：DcfManagerにアクセス要求を出す
  1. 送信可能なパケットが存在する場合（キューにパケットまたは再送パケットが存在）かつMacLowにアクセスしていない場合
     1. DcfManagerのRequestAccess (m_dcf)を呼び出す

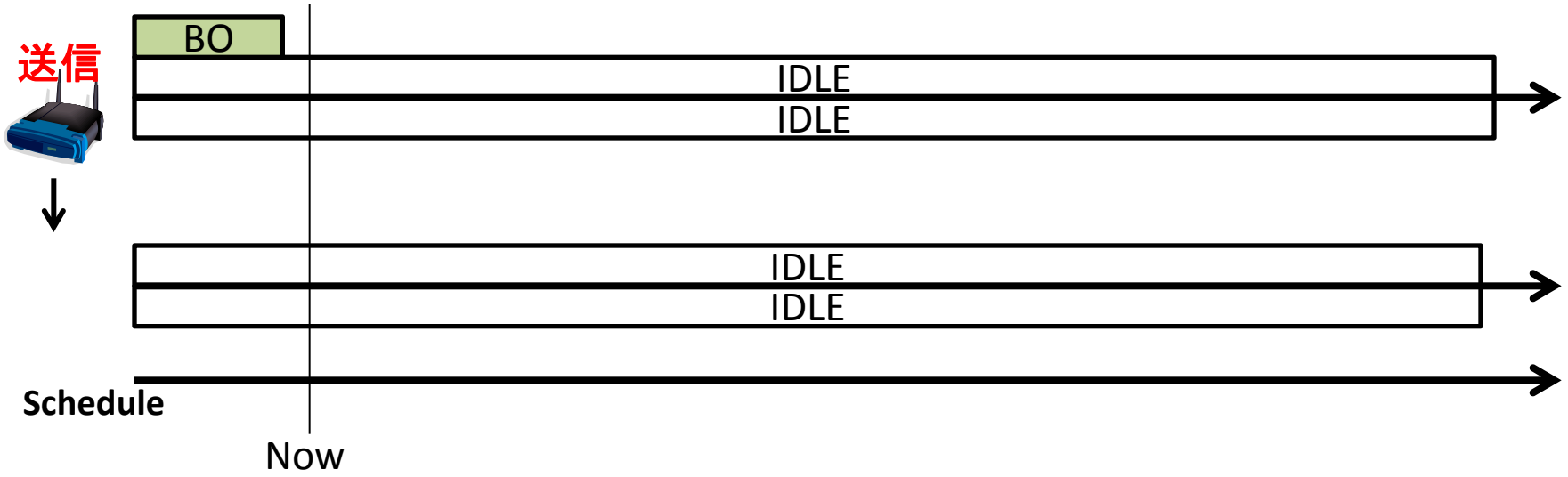


DcaTxop：送信

DcfManager

RequestAccess (state)

- 引数：
  - state: DcfState
- 仕様：MacLowにアクセス可能な場合，DcaTxopにMacLowのアクセス許可を与える
  1. この関数の呼び出し時間がBO終了時間を超えていた場合
     1. DcaTxopのNotifyAccessGranted ()関数を呼び出す

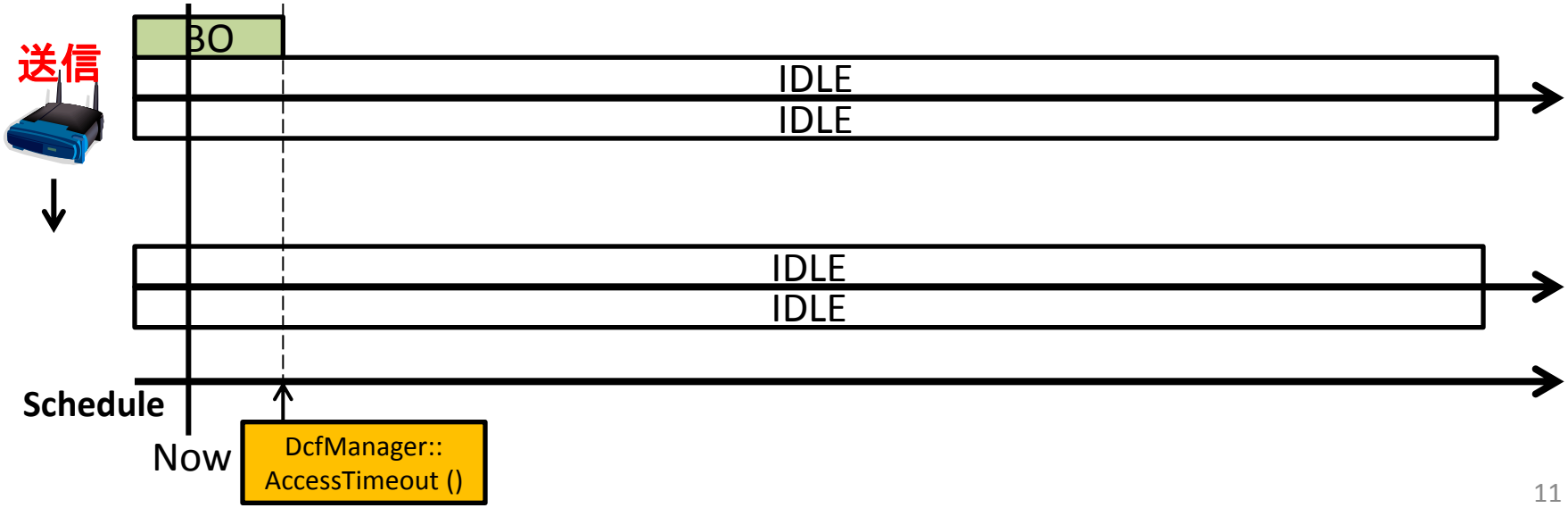


DcaTxop：送信

DcfManager

RequestAccess (state)

- 引数：
  - state: DcfState
- 仕様：MacLowにアクセス可能な場合，DcaTxopにMacLowのアクセス許可を与える
  1. この関数の呼び出し時間がBO終了時間を超えていた場合
     1. DcaTxopのNotifyAccessGranted ()関数を呼び出す
  2. この関数の呼び出し時間がBO終了時間を超えていない場合
     1. AccessTimeout ()関数をBO終了時間にスケジューリング

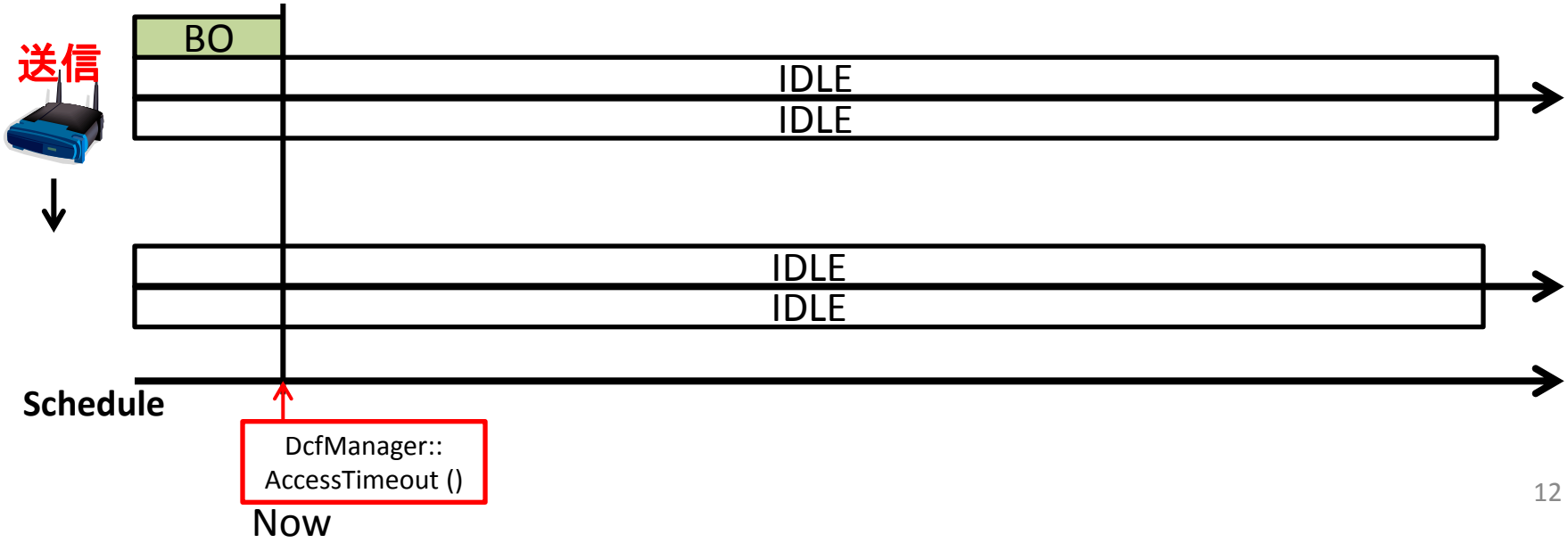


DcaTxop：送信

DcfManager

AccessTimeout () ***※RequestAccess ()関数とほとんど一緒の役割***

- 仕様：MacLowにアクセス可能な場合，DcaTxopにMacLowのアクセス許可を与える
  1. この関数の呼び出し時間がBO終了時間を超えていた場合
     1. DcaTxopのNotifyAccessGranted ()関数を呼び出す
  2. この関数の呼び出し時間がBO終了時間を超えていない場合
     1. AccessTimeout ()関数をBO終了時間にスケジューリング

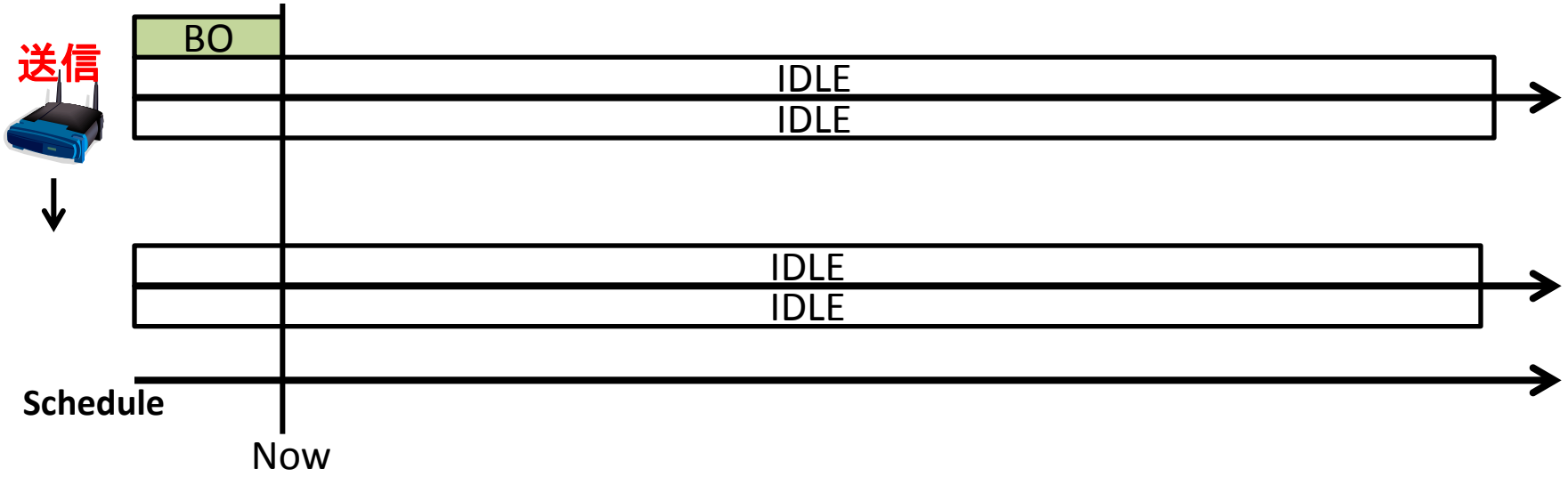


DcaTxop：送信

DcaTxop

NotifyAccessGranted ()

- 仕様：Mac層のパラメータを生成して，MacLowにパケットを渡す
  1. m_currentPacketにパケットを代入
  2. MacLowTransmissionParameter paramを生成（RTS，ACKがいるかどうかなどの情報が入る）
  3. MacLowのStartSecondaryTransmission (m_currentPacket, &m_currentHdr, params, m_transmissionListener)を呼び出す



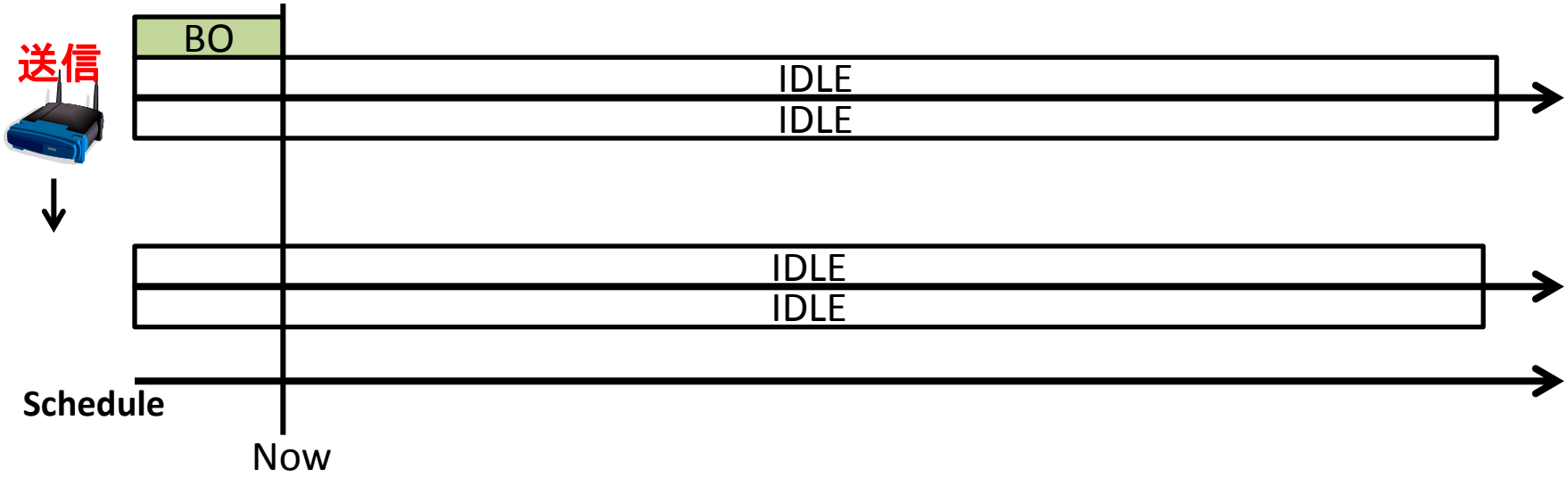
MacLow：送信

MacLow

StartTransmission (packet, hdr, params, listener)

- 引数
  - packet: 送信パケット
  - hdr: 送信ヘッダ
  - params: MACのパラメータ（DATA，ACK，RTS，CTSを判別するのに利用）
  - listener: DcaTxopのリスナー（ACKやCTSのタイムアウトを通知するのに利用）
- 仕様：引数をメンバ変数に格納，データタイプに基づき送信
  1. 引数で与えられた変数をメンバ変数に格納
  2. paramsに基づきDATA，ACK，RTS，CTSを判別して送信
  3. DATAの場合はSendDataPacket ()関数を呼び出す

・m_currentPacket：送信パケット
・m_currentHdr: 送信ヘッダ
・m_listener： DcaTxopのリスナー
・m_txParams： MAC層のパラメータ

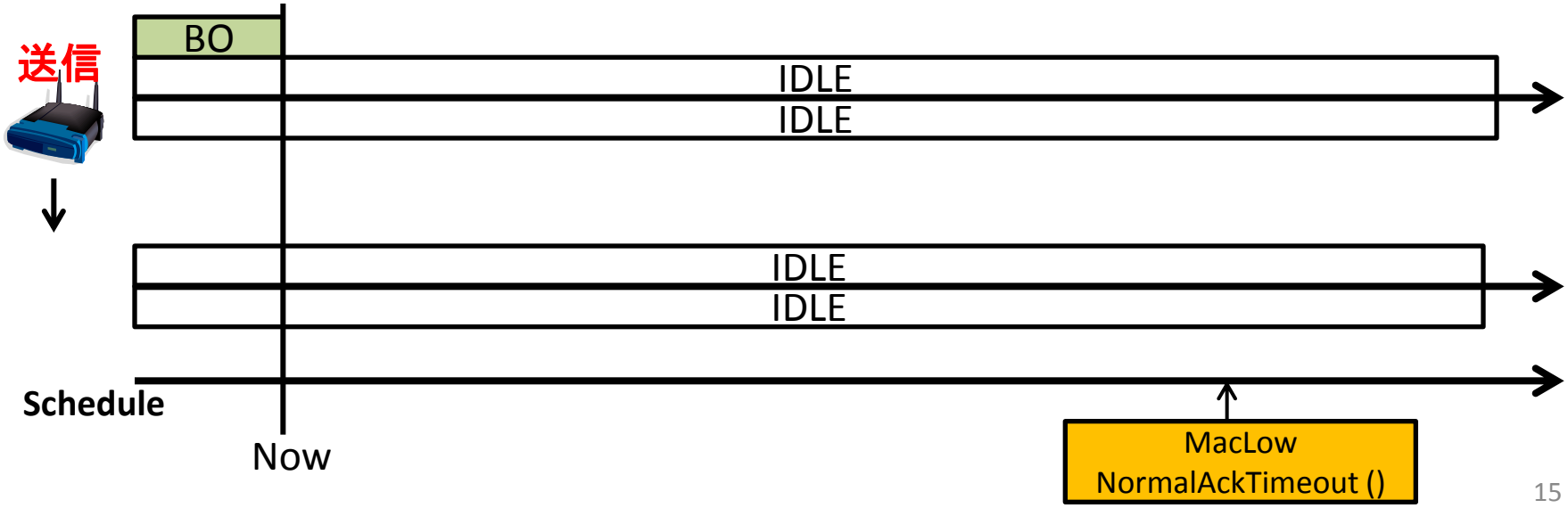


MacLow：送信

MacLow

SendDataPacket ()

- 仕様：物理層のデータ集合を生成，MACフレームの生成
  1. dataTxVectorを生成：m_currentHdrから物理層のデータの集合（レート情報，送信電力情報，送信アンテナ数など）を生成
  2. preambleを生成：Long Preamble，Greenfield Preambleなど
  3. ACKのタイムアウトをスケジューリング
  4. durationを求める：m_currentHdrのTypeとフラグメント化の情報から求める
  5. m_currentPacketにm_currentHdrとFCS（Frame Check Sequence）の付加
  6. ForwardDown (m_currentPacket, m_currentHdr, dataTxVector, preamble)を呼び出す



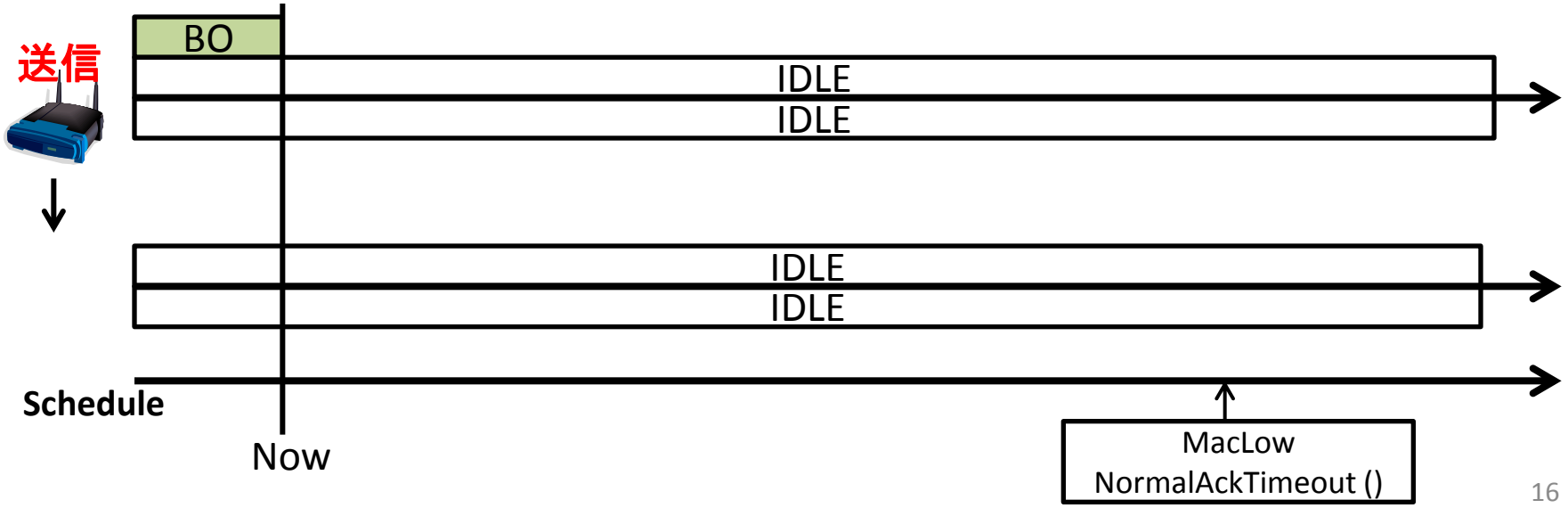
MacLow：送信

MacLow

ForwardDown (packe, hdr, txVector, preamble)

- 仕様：物理層のSendPacket ()関数を呼び出す
  1. 物理層のSendPacket (packet, txVector.GetMode(), preamble, txVector)を呼び出す

YansWifiPhy::SendPacket (packet, hdr, txMode, preamble, txVector)



YansWifiPhy : 送信

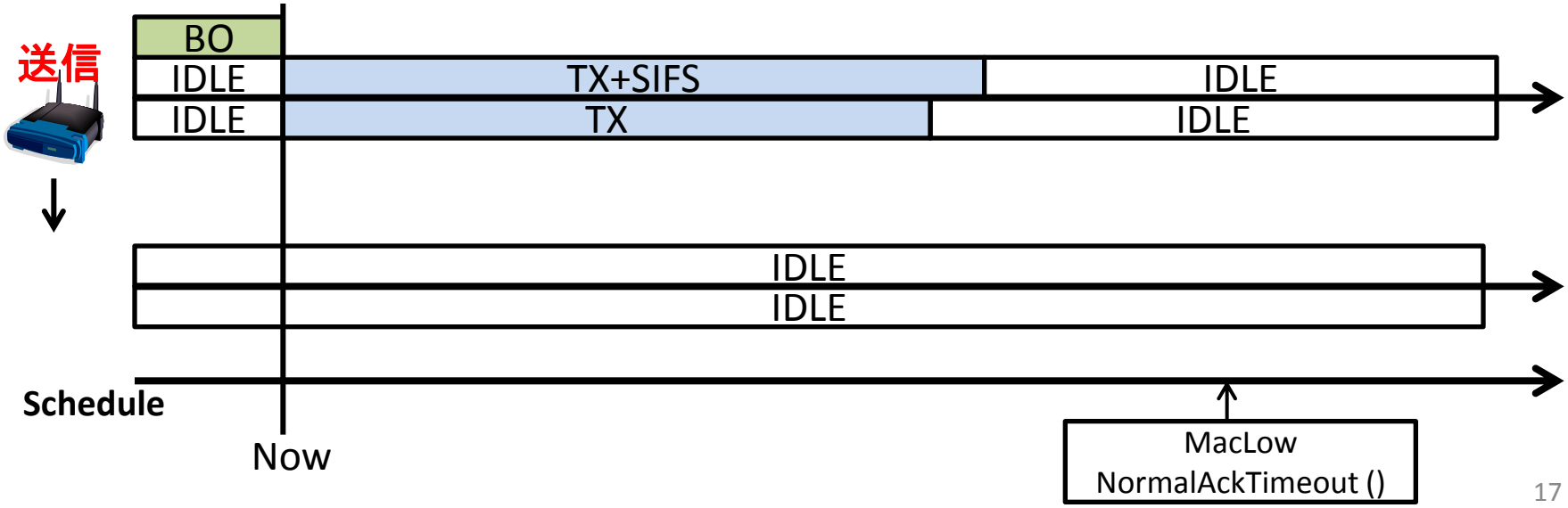
YansWifiPhy

SendPacket (packe, hdr, txMode, preamble, txVector)

- 引数
  - txMode: バンド幅，コードレート，変調方式などの情報を持つ
- 仕様 : WifiPhyStateHelperのステータスを変化，チャネルクラスにのSend ()関数を呼び出す
  1. WifiPhyStateHelperのSwitchTx ()関数を呼び出してステータスを変化
  2. チャネルクラスのSend (this, packet, GetPowerDbm (txVector.GetTxPowerLevel()) + m_txGainDb, txVector, preamble)を呼び出す

送信ゲイン

YansWifiChannel::Send (sender, packet, txPowerDbm, txVector, preamble)

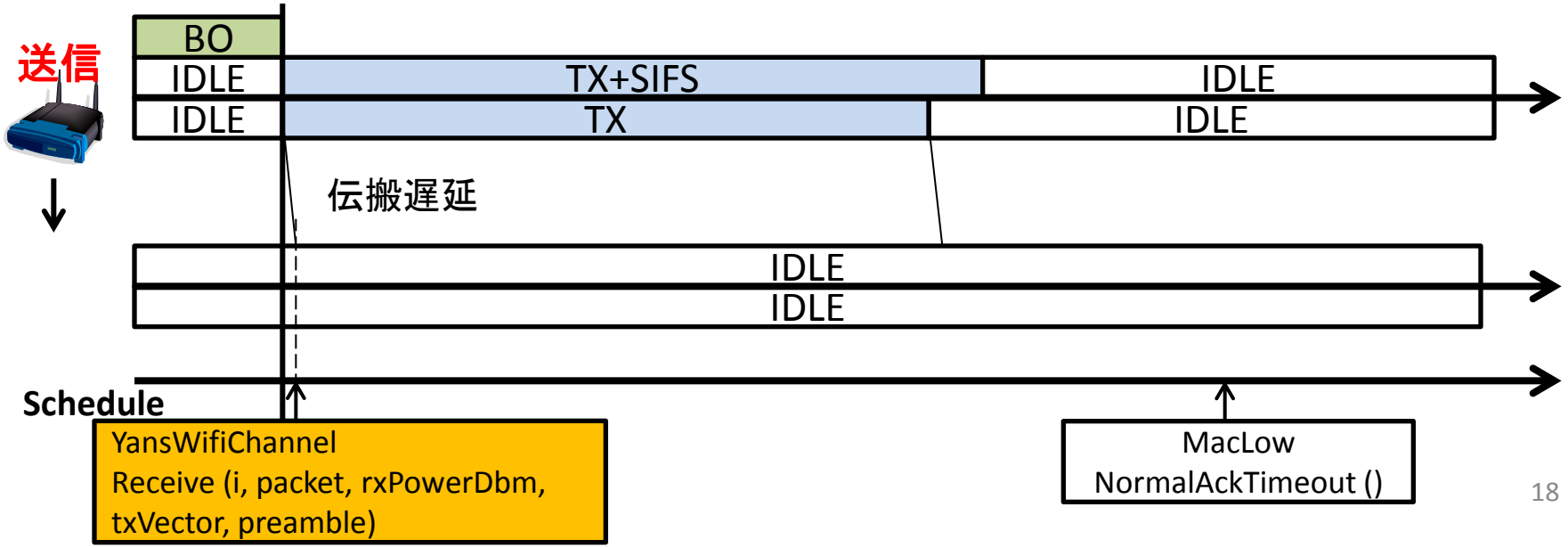


YansWifiChannel：送信

YansWifiChannel

Send (sender, packet, txPowerDbm, txVector, preamble)

- 引数
  - txPowerDbm: 送信電力
- 仕様：遅延，伝搬損失の算出，算出した結果に基づきReceive()関数をスケジューリング
  1. senderMobilityを取得：senderからモビリティ情報（位置情報等）を取得
  2. receiverMobilityを取得：受信ノードのモビリティ情報を取得
  3. 取得したモビリティ情報から遅延（delay），受信電力（rxPowerDbm）を計算
  4. Receive（受信と送信で共通のインデックス，パケット，rxPowerDbm，txVector，preamble）関数を伝搬遅延後にスケジューリング



YansWifiChannel : 受信

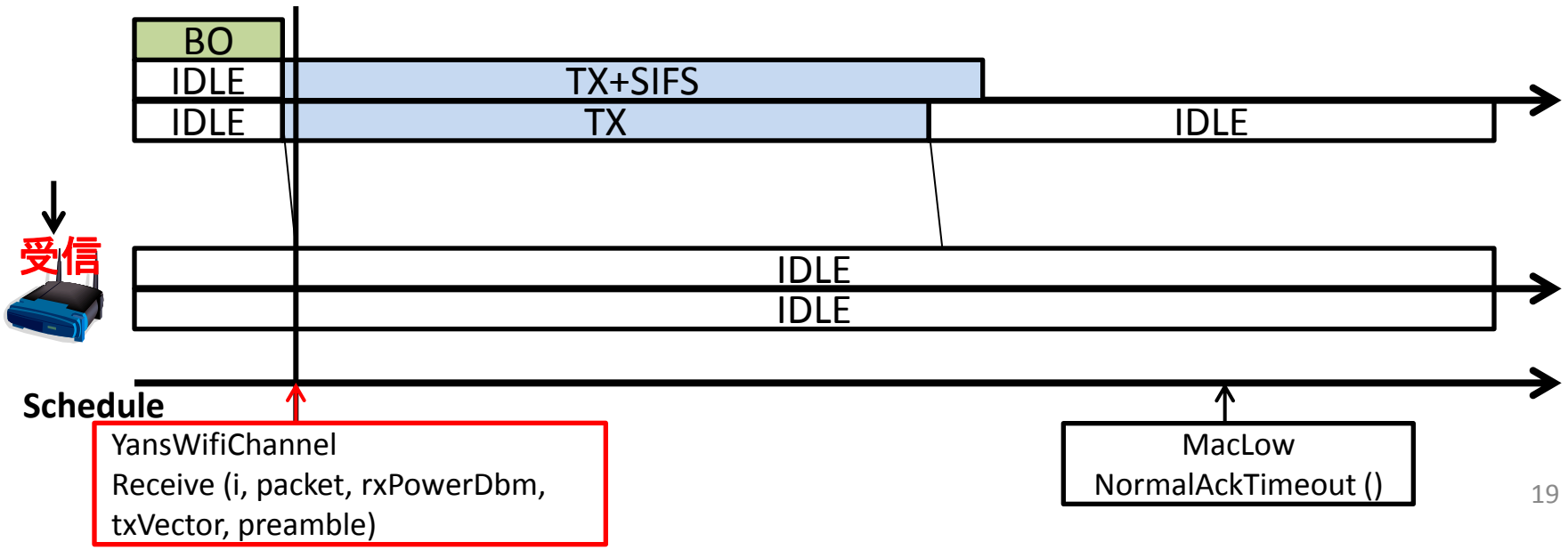
YansWifiChannel

Receive (i, packet, rxPowerDbm, txVector, preamble)

- 仕様 : 物理層の受信処理
  1. 共通の番号iを用いてStartReceivePacket (packet, rxPowerDbm, txVector, preamble)を呼び出す

↓

YansWifiPhy::StartReceivePacket (packet, rxPowerDbm, txVector, preamble)

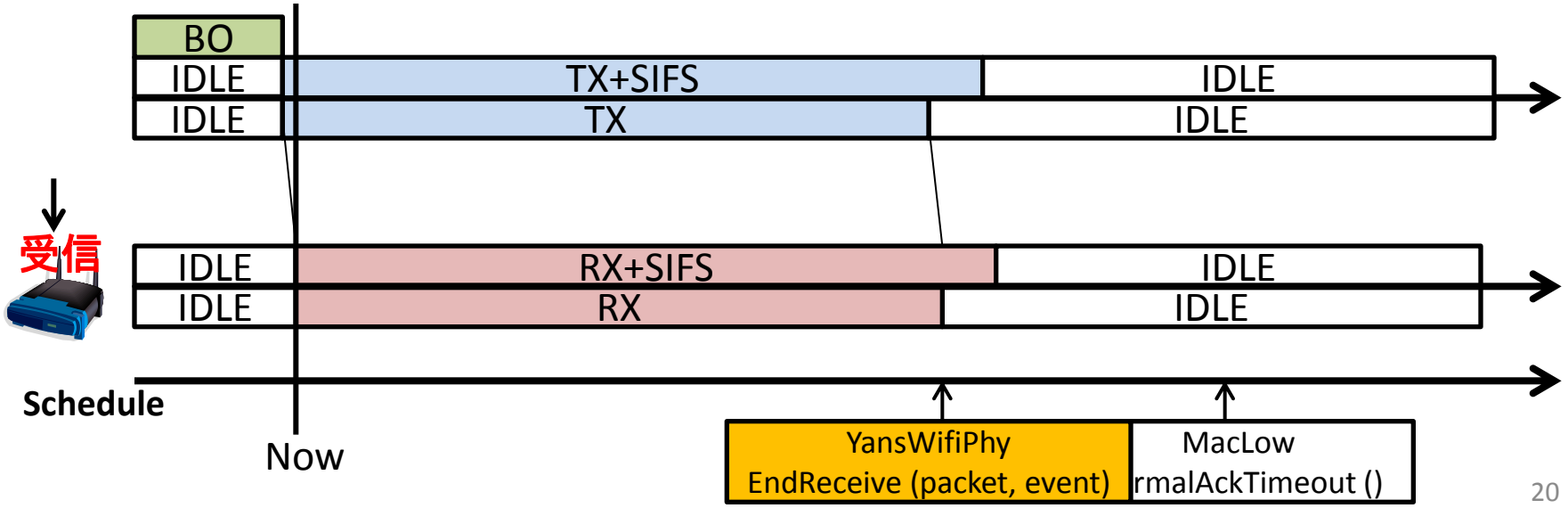


YansWifiPhy : 受信

YansWifiPhy

StartReceivePacket (packet, rxPowerDbm, txVector, preamble)

- 仕様 : パケットの受信開始時の処理
  1. rxDuration算出: パケットサイズ，txVector（伝送レート），preambleから受信時間を計算
  2. InterferenceHelperにeventを追加（SNR・PERの計算に利用 ※ 後で説明）
  3. 物理層の状態がCCA_BUSYまたはIDLE状態の場合かつ受信電力が受信スレッショルドを上回っていた場合
     1. WifiPhyStateHelperのSwitchRx()関数を呼び出し状態を変える
     2. EndReceive (packet, event)をスケジューリング



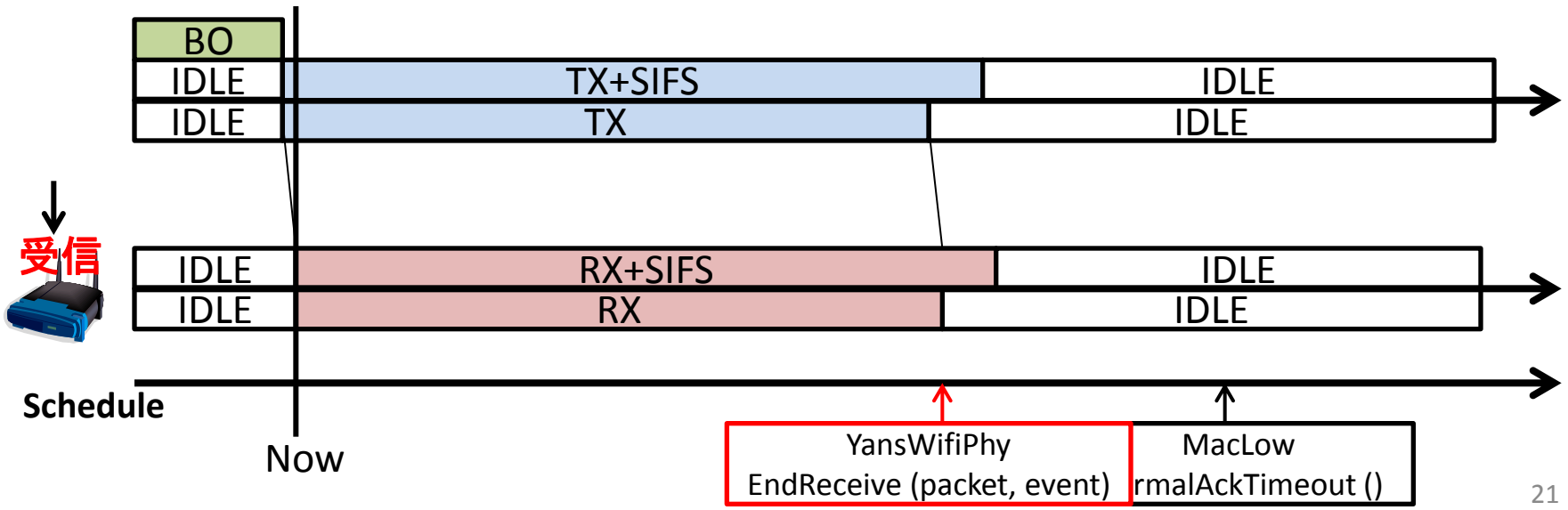
YansWifiPhy：受信

YansWifiPhy

EndReceive (packet, event)

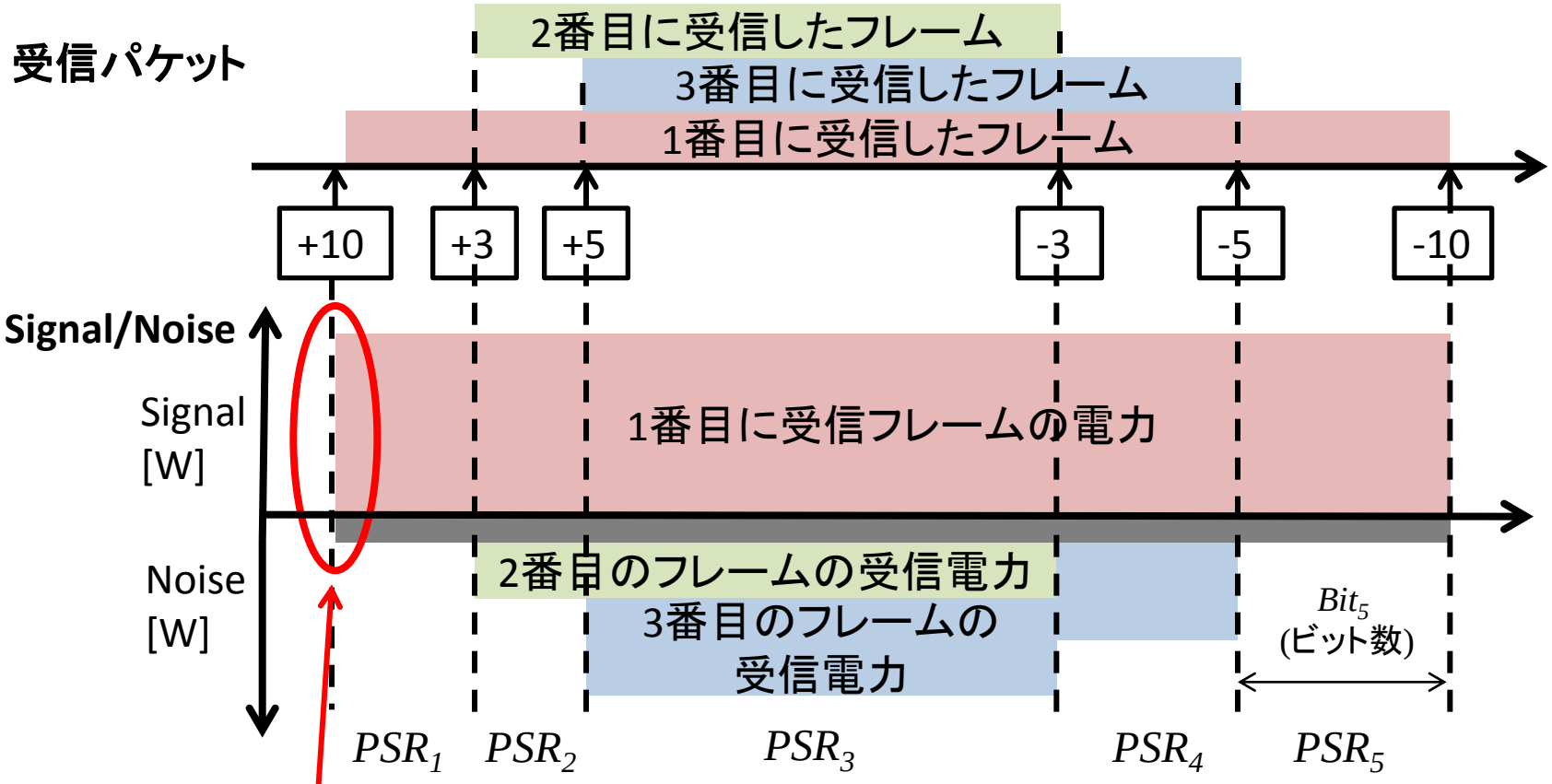
- 仕様：PSRを用いた受信判定処理
  1. SNR（Signal to Noise Ratio）とPSR（Packet Success Rate）を求める
  2. random値を生成：[0-1)のランダム値
  3. random ＞PSR の場合（受信成功）
     1. MacLowのReceiveOk (packet, rxSnr, txMode, preamble)を呼び出す
  4. random <= PSRの場合（受信失敗）
     1. MacLowのReceiveError (packet, rxSnr, preamble)を呼び出す

MacLow::ReceiveOk (packet, rxSnr, txMode, preamble)



InterferenceHelper
~SNR・PERの求め方~

- SNRは受信開始時のSignalとNoiseの値から算出
- PERは電力が変動する区間ごとに算出



$$SNR = \frac{Signal}{Noise}$$

$$PER = 1 - \left(\prod_l PSR_l\right)$$

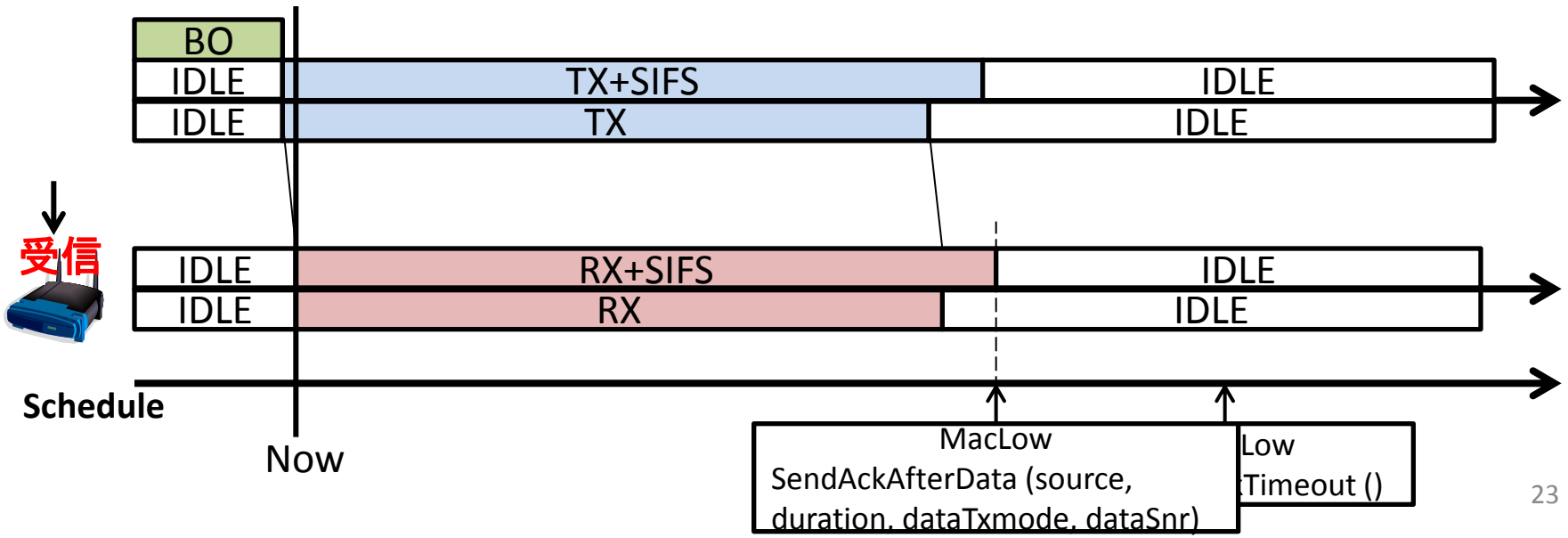
PSR_5= Bit_5と変調方式，SNRから理論値で算出

MacLow：受信

MacLow

ReceiveOk (packet, event, txMode, preamble)

- 仕様：受信後の処理，DATAの場合SIFS後にACKを返信
  1. hdrを生成：packetからMACのヘッダを取り出す
  2. hdrからDATA，ACK，RTS，CTSを判断する
  3. DATAの場合かつ自身のアドレスが指定されている場合
     1. SendAckAfterData（送信元アドレス，hdrのDuration，txMode，rxSnr）をSIFS時間後に実行されるようにスケジューリング



MacLow：ACK送信

MacLow

SendAckAfterData (source, duration, dataTxMode, dataSnr)

- 仕様：ACKの送信
  1. ackTxVectorを求める：sourceとdataTxModeから求める（レートや送信電力等）
  2. packetを生成
  3. ack（ACKのヘッダ）を生成：
  4. packetにackとFCSを付加
  5. preambleを生成：
  6. ForwardDown (packet, &ack, ackTxVector, preamble)を呼び出す

ForwardDown()関数の呼び出し後はDATA送信と同様に処理