SmartMesh IP
アプリケーション・ノート

目次

1 改訂履歴

リビジョン	日付	概要
1.	2013/03/18	初期リリース
2	2013/07/18	「パケット ID の概要と必要な理由」アプリケーション・ノートを追加
3	2013/10/22	軽微な修正
4	2014/01/30	「重複ネットワーク」アプリケーション・ノートを追加。「メッシュ内メッシュの構築」アプリケーション・ノートを修正してサンプル実装を反映
5	2014/04/04	Q の計算方法を明確化
6	2014/04/30	「深い IP ネットワークの構築」アプリケーション・ノートを更新
7	2014/10/28	「SmartMesh のセキュリティ」アプリケーション・ノートの軽微な変更。「SmartMesh IP のデータ発行」アプリケーション・ノート内で起動イベント・フラグを明確化。「ネットワーク性能とデバイス性能の評価方法」アプリケーション・ノート内でアイドル電流の測定における APIExplorer への接続の必要性を明確化
8	2015/04/21	「干渉の影響の特定および軽減」、「SmartMesh IP のデータ発行」、「ネットワーク性能とデバイス性能の評価方法」アプリケーション・ノートを更新
9	2015/12/03	「モート・パラメータのリモート読み出し方法」アプリケーション・ノートを追加。ハードウェアの説明を更新。
10	2016/05/25	「SmartMesh IP ネットワークでの 6LoWPAN とルーティング」、「OTAP アプリケーションの構築」アプリケーション・ノートを追加
11	2016/07/08	「ネットワーク健全性の予測」アプリケーション・ノートの名称を「CLI を使用した組込みマネージャのネットワーク健全性の予測」に変更して例を追加
12	2016/07/28	「ネットワーク性能とデバイス性能の評価方法」アプリケーション・ノートの電力セクションを更新
13	2016/09/19	「バックグラウンド・ノイズの監視」アプリケーション・ノートを追加
14	2016/11/01	いくつかの古いアプリケーション・ノートを削除、組込みマネージャの配置と VManager の配置を区別するため、いくつかのアプリケーション・ノートの名前を変更して改定
15	2017/01/30	関連資料ページを追加

2 アプリケーション・ノートの一覧

本書では、SmartMesh IP 製品ファミリに関する一連のアプリケーション・ノートについて記載しています。

- ネットワーク性能とデバイス性能の評価方法 - 性能測定基準についてホストを測定する方法。
- 組込みマネージャのサブスクライブ API フィルタの使用法 - マネージャ通知をモニタする方法。
- 組込みマネージャでの SmartMesh IP ネットワークの健全性のモニタ - ネットワークが良好に機能していることを確認する自動検査。
- SmartMesh IP のデータ発行 - モートの API を使用してデータを送信する方法の具体的な説明。
- 組込みマネージャのネットワークでの通知の管理 - 通知をオフにすることによって電力を節約する状態マシン。
- 組込みマネージャの通電バックボーンを使用した待ち時間の改善 - いくつかのバックボーンの使用事例の説明。
- 深い IP ネットワークの構築 - 最大 32 ホップの深さのネットワークを使用するための設定。
- 重複ネットワーク - 同じ無線空間に安全に共存できるモート数の計算。
- モート・パラメータのリモート読出し方法 - netGetParameter コマンドを使用して無線でモート・パラメータを問い合わせる方法。
- SmartMesh IP ネットワークでの 6LoWPAN とルーティング - IPv6 関連ヘッダのネットワークでの使用に関する説明。
- VManager の OTAP アプリケーションの構築 - VManager を使用してモートのコードを更新する OTAP の実装ガイド。
- 組込みマネージャの OTAP アプリケーションの構築 - 組込みマネージャを使用してモートのコードを更新する OTAP の実装ガイド。
- CLI を使用した組込みマネージャのネットワーク健全性の予測 - 初期の配置後の評価のヒント。
- バックグラウンド・ノイズの監視 - チャンネル固有の健全性レポートを使用したワイヤレス干渉の診断。

以下のアプリケーション・ノートは、SmartMesh IP ファミリと WirelessHART ファミリの両方に当てはまります。製品間の違いがある場合は、違いが強調されます。

- 配置の計画 - ネットワークを配置する前の概略の検討。
- よくある問題と解決策 - トラブルシューティングに関する一般的なアドバイス。
- プロビジョニング係数の変更によるマネージャ・スループットの増加 - 帯域幅をモートごとに低くすることにより、安定性の高い状態での総トラフィック増加をサポート。
- 輻輳したネットワークのデバッグ - モート・キューが満杯の場合のネットワーク動作に対する影響の理解と、そのような状況の改善に関するヒント。
- 干渉の影響の特定および軽減 - 干渉に関係がある問題を見つけるためのネットワーク統計情報のグラフ表現。
- 正確なタイムスタンプの取得 - ネットワーク時刻の同期を使用してパケットに正しいタイムスタンプをする方法。
- 複数のマネージャを使用した大規模ネットワークの構築 - 1 つのマネージャのモート限度を超える配置に関する検討事項。

- SmartMesh Power and Performance Estimator の使用 - 様々なネットワークで電力および待ち時間を予測するのにスプレッドシートを使用する場合の例。
- モートの移動によって予想されること - 同じネットワーク内の異なる場所へ移動するモートの動作に関する詳細。
- ネットワーク間でのモートの移動 - 様々なネットワーク間でモートを移動する方法に関する詳細。
- 境界内データ待ち時間に関するネットワークの構成 - ネットワークへの帯域幅の追加によるパケット待ち時間の短縮。
- ネットワークの共存 - 複数のネットワークが同じ無線空間で動作できる機能。
- ジョイン・デューティ・サイクルの選択方法 - 電力と参加速度とのトレードオフに関する検討。
- SmartMesh のセキュリティ - すべての SmartMesh ネットワークで使用されるセキュリティ機能の説明。
- TestRadio コマンドの使用 - API を使用して無線性能のテストまたは認証を行う方法の説明。
- ピーク時の平均電流を制限するための最良実施例 - フラッシュの起動または書込み時に平均電流を少量に維持するために従うガイドライン。
- パイロット・ネットワーク評価の方法 - パイロット・ネットワークの配置および統計情報収集の概要。
- パケット ID の概要と必要な理由- センサー・プロセッサによる信頼性の高い呼び出しと応答を実現するために使用される単一ビットに関する詳細。

3 関連資料

SmartMesh IP ネットワーク向けに以下の資料が提供されています。

Starter Kit のクイック・ガイド

- SmartMesh IP Easy Start Guide - 基本的なインストール方法とネットワークの動作確認テストについて説明しています。
- SmartMesh IP Tools Guide - インストールのセクションではシリアル・ドライバのインストール手順について説明しており、Easy Start Guide やその他のチュートリアルで使用されるサンプル・プログラムも含まれています。

ユーザ・ガイド

- SmartMesh IP User's Guide - ネットワーク概念についての説明と、モートおよびマネージャの API を使用して特定のタスク（データ送信や統計情報の収集など）を実行する方法について説明しています。この資料は、API ガイドを使用するための予備知識を提供します。

デバイスの対話操作用インターフェース

- SmartMesh IP Embedded Manager CLI Guide - マネージャとやり取りするために使用します（クライアントの開発中やトラブルシューティングなど）。このガイドは、CLI の接続とそのコマンド・セットについて説明しています。
- SmartMesh IP Embedded Manager API Guide - プログラムを使用してマネージャとやり取りするために使用します。このガイドは、API の接続とそのコマンド・セットについて説明しています。
- SmartMesh IP Mote CLI Guide - モートとやり取りするために使用します（センサー・アプリケーションの開発中やトラブルシューティングなど）。このガイドは、CLI の接続とそのコマンド・セットについて説明しています。
- SmartMesh IP Mote API Guide - プログラムを使用してモートとやり取りするために使用します。このガイドは、API の接続とそのコマンド・セットについて説明しています。

ソフトウェア開発ツール

- SmartMesh IP Tools Guide - SmartMesh SDK に含まれる各種の評価および開発サポート・ツールについて説明しています。モートおよびマネージャ API の使用とネットワークの視覚化のためのツールを含みます。

アプリケーション・ノート

- SmartMesh IP Application Notes - SmartMesh IP ネットワーク固有の各種トピックと、SmartMesh ネットワーク全般に当てはまるトピックが含まれています。

新規設計の開始時に役立つ資料

- LTC5800-IPM SoC またはこれに基づく modules のデータシート。
- LTC5800-IPR SoC またはこれに基づく embedded managers のデータシート。

- モート／マネージャ SoC 用または module 用の Hardware Integration Guide - 設計に SoC またはモジュールの統合を盛り込むためのベスト・プラクティスを提供しています。
- 組込みマネージャ用の Hardware Integration Guide - 設計する際に組込みマネージャを統合させるためのベスト・プラクティスを提供しています。
- Board Specific Integration Guide - SoC モートおよびマネージャに対するデフォルトの I/O 設定方法と、「ヒューズ表」を使用した水晶発振器のキャリブレーション情報について説明しています。
- Hardware Integration Application Notes - SoC 設計チェックリスト、アンテナ選定ガイドなどを含みます。
- ESP Programmer Guide - DC9010 Programmer Board と、デバイスへのファームウェアのロードに使用する ESP ソフトウェアのガイドです。
- ESP ソフトウェア - モートまたはモジュールにファームウェア・イメージをプログラミングするために使用します。
- Fuse Table ソフトウェア - Board Specific Configuration Guide で説明されているヒューズ表を作成するために使用します。

その他の役立つ資料

- SmartMesh 技術文書で使用されるワイヤレス・ネットワークの用語については、SmartMesh IP User's Guide を参照してください。
- よくある質問の一覧。

4 アプリケーション・ノート：ネットワーク性能とデバイス性能の評価方法

4.1 概要

SmartMesh IP starter kit を入手したばかりですが、これからどうしますか。このアプリケーション・ノートでは、SmartMesh ネットワークの特定の性能特性を評価するために、組込みマネージャを使用して実行できるいくつかのテストについて説明します。ここでは、マネージャおよびモートと通信するために、SmartMesh IP SDK Python ツールと必要な FTDI ドライバをインストール済みであることを前提としています。「Manager and Mote CLI and API」の資料を参照することも前提としています。

最初にすることは、SmartMesh IPEmbedded Manager CLI Guide の Introduction（はじめに）に説明されているように、マネージャに接続し、マネージャの CLI にログインすることです。

```
> login user
```

モートの CLI にアクセスして構成するために、DC9006 Eterna インターフェース・カードを 1 つ以上の DC9003A-B モートに接続することも必要になります。

4.2 参加動作

モートが参加する速さはどのくらいでしょうか。モートの参加速度は、以下の 6 つの条件の関数です。

- 通知送信率 - ネットワーク内のモートが通知を送信する割合。モート密度以外はこれをほとんど制御できません。あるいは、マネージャで明示的にオフにします。
- ジョイン・デューティ・サイクル - 探索モートがネットワークの検出に費やす時間とスリープ状態の時間との比。
- モート加入状態のマシン・タイムアウト - SmartMesh IP ではこれらのタイムアウトを制御できません。
- 下り帯域幅 - モートがネットワークに同期された状態から**稼働**状態（つまり、データを送信可能な状態）にどの程度素早く移行できるかに影響します。
- モートの数 - 限られたリソースを求めて多くのモートが同時に参加しようとすると、衝突により参加の速度が低下します。
- 経路安定性 - ほとんどあるいはまったく制御できないもの。経路安定性は、1 組のモート間での送信パケットに対する（アクノリッジを返された）成功パケットの比率です。

参加処理は、探索、同期、メッセージ交換という 3 つの段階に分れます。

- 探索時間は、通知送信率、経路安定性、およびモートジョイン・デューティ・サイクルによって決まります。この時間はジョイン・デューティ・サイクルに大きく依存し、数十秒から数十分になることがあります。
- 同期はモート状態のマシン・タイムアウトによって決まります。この時間は SmartMesh IP ではわずか数秒です。

- メッセージ交換は下り帯域幅と（下り帯域幅を求めて競合する）モートの数によって決まります。これはモート当たり最小で約 10 秒で、より低速化する可能性があります。

したがって、例えばネットワークに素早く参加するには、ジョイン・デューティ・サイクルを最大に設定することになるように思えます。しかし、そうすると多くのモートが下り帯域幅を求めて競合することになる可能性があり、ネットワークはより低めの設定値にすることで素早く形成できます。以下の実験では、探索段階およびメッセージ交換段階を制御するための「調整つまみ」を調べます。

4.2.1 探索時間と平均電流のトレードオフ

同期していないモートはマネージャやその他のモートからの通知を受信して、ネットワークと同期します。受信に費やす時間とスリープ状態の時間との比を**ジョイン・デューティ・サイクル**と呼びます。ジョイン・デューティ・サイクルは、0（0.2%）～255（ほぼ 100%）に設定できる 1 バイトのフィールドです。スタータ・キットのデフォルト値は 100%つまり 255 に設定されています。設定値を低めにすると探索時間が長く平均電流が少なくなる一方で、設定値を高めにすると探索時間は短くなるものの平均電流は増加します。少なくとも 1 つのモートがそのモートの近くに通知を送信しているとすると、参加に割かれる**エネルギー**は同じです。ジョイン・デューティ・サイクルは平均**電流**だけに影響します。モートが隔離されている場合、マネージャがオンでない場合、または通知がオフになっていた場合、モートはネットワークを探してバッテリを使い果たす可能性があります。ジョイン・デューティ・サイクルによって、ネットワークが存在する場合の速度とネットワークが存在しない場合のエネルギー使用量とのトレードオフを制御できます。

モートが同期と参加要求送信を行う時間は、ネットワークに参加するモートの最初の明らかな兆候です。次のような `trace` CLI コマンドを使用してマネージャのモート状態トレースを起動できます。

```
> trace motest on
```

マネージャは、モートの参加要求を確認すると、**Negotiation1**（Negot1）状態であるとモートにマークを付けます。

ジョイン・デューティ・サイクルの効果を確認するには、次のようにします。

- マネージャ CLI に接続し、前述したようにモート状態トレースをオンします。
- Eterna インターフェース・カード（DC9006）を使用してモート CLI に接続します。
- モートをリセットします。モートはリセット・メッセージを出力します。

```
> reset
SmartMesh IP mote, ver 1.1.0.32 (0x0)
```

これにより、モートはデフォルトのデューティ・サイクルでネットワークの探索を開始します。モートの状態が変わると、ミリ秒単位のタイムスタンプの後に状態という形式で一連の通知が表示されます。

```
52727 : Joining
56227 : Connected
60121 : Active
```

モートが同期してその参加要求を送信するのにかかる時間は、平均で 10 秒未満です。これを数回繰り返して(毎回リセットして)、同期時間の分布を調べたい場合があります。

- モートをリセットし、モート CLI を使用してジョイン・デューティ・サイクルを 5%(0.05 * 255 = 13)に設定します。これはモートの `mset joindc` CLI コマンドによって制御します。ジョイン・デューティ・サイクルは SmartMesh IP モートでは不揮発性です。リセットや電源の入れ直しをしても持続するので、この手順を実行する必要があるのは、ジョイン・デューティ・サイクルを変更するたびに 1 回だけです。モートが **Negotiating1** 状態に遷移する所要時間を測定し、前述したようにこの測定を繰り返します。所用時間は平均で約 3 分になります。

```
> mset joindc 13
```

- モートをリセットし、モート CLI を使用してジョイン・デューティ・サイクルを 25%(64)に設定します。モートが **Negotiating1** 状態に遷移する所要時間を測定します。所用時間は平均で約 30 秒になります。

```
> mset joindc 64
```

注記:マネージャは、以前は稼働状態だったモートからの参加要求を認識すると、そのモートにまず **Lost** とマークを付けてから connected(接続)となるように進めます。これは以下のように表示されます。

```
264504 : Mote #2 State:    Lost  -> Negot1
266178 : Mote #2 State:  Negot1  -> Negot2
```

4.2.2 参加時間の測定

マネージャがいくつかのパケットを送信し、モートがアクノリッジを返すことにより、モートはデータの送信を開始できる **Operational**（Oper、稼働）状態に遷移します。マネージャのモート状態トレース CLI コマンド（このトレースは前述のテストから引き続きオンのまま）を使用して、これらの状態遷移を確認し、その所要時間を測定できます。タイムスタンプの単位はミリ秒です。モートをリセットして、モートが **Operational** に遷移するのを注視します。以下のトレースの各行は、モートとマネージャの間の 1 回のハンドシェイクを表します。

```
117757 : Created mote 2
117765 : Mote #2 State:    Idle  -> Negot1

119853 : Mote #2 State:  Negot1  -> Negot2

122366 : Mote #2 State:  Negot2  -> Conn1

125222 : Mote #2 State:   Conn1  -> Conn2

127127 : Mote #2 State:   Conn2  -> Conn3

129185 : Mote #2 State:   Conn3  -> Oper
```

この場合は遷移に約 11.5 秒かかっています。

4.2.3 参加時の下り帯域幅の影響

マネージャがパケットを送信できる速度は、下りスロットフレーム・サイズの関数です。マネージャはスロットフレーム・サイズに関係なく、同じ数の下りリンクを割り当てるので、長さが 4 倍のスロットフレームは、帯域幅が 1/4 になります。これを図 1 に示します。100 スロットのスロットフレームでのスロット 1 は、200 スロットのスロットフレームでの同じスロット（スロット 1）より 2 倍の頻度で繰り返されます。デフォルトでは、マネージャが「高速」256 スロットのスロットフレームにネットワークを構築し、1 時間後に「低速」スロットフレームに遷移します。低速スロットフレームのスロット数は、256（デフォルト）、512、1024 のいずれでもかまいません。この値は **dnfr_mult**（下りスロットフレーム乗数）パラメータを介して設定されます。より長いフレームは単位時間当たりの下り受信回数が少ないことを意味するので、フレームが長いとすべてのモートの平均電流が低下しますが、後で追加されるモートに備えて参加処理（やその他の下りデータ）も低速になります。これはネットワークに同期するための所要時間（**Negotiating1** 状態までの時間）には影響しませんが、その他の状態遷移の間隔が一定になります。

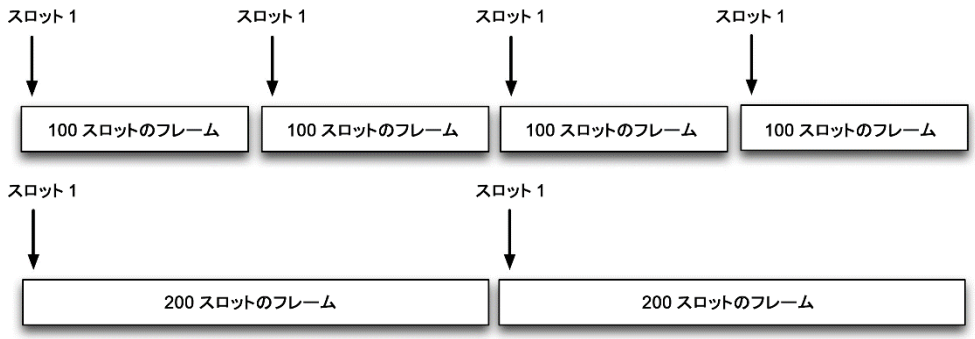


図 1 - 200 スロットフレームの場合より 100 スロットフレームの場合の方がスロット 1 が頻繁に繰り返される

これらは公称サイズであることに注意してください。実際のスロットフレーム・サイズは多少ランダム化されており、複数のネットワークが部分的に重なっている場合、共存状態が改善されます。本書では全体を通じて公称サイズを指します。

より長い下りスロットフレームの影響を調べるには、以下のようにします。

- マネージャ CLI により、通常の「高速」スロットフレームが使用されていることを確認します。下りスロットフレーム乗数（**dnfr_mult**）を示す行を探します。

```
> show config
netid = 1229
txpower = 8
frprofile = 1
maxmotes = 17
basebw = 9000
dnfr_mult = 1
numparents = 2
cca = 0
channellist = 00:00:7f:ff
autostart = 1
locmode = 0
bbmode = 0
bbsize = 1
license = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
ip6prefix = fe:80:00:00:00:00:00:00:00:00:00:00:00:00:00:00
ip6mask = ff:ff:ff:ff:ff:ff:ff:ff:00:00:00:00:00:00:00:00
radiotest = 0
bwmult = 300
```

- ここで下りスロットフレーム乗数を調整して、1024 スロットのスロットフレームが得られるようにします。乗数を 4 に設定すれば、4*256 スロットのスロットフレームが得られます。以下に示す config パラメータは不揮発性です。

```
> set config dnfr_mult 4
```

- より長いスロットフレームの使用を押し進めます。

```
> exec setDnframe normal
Start Global Unicast Command setDnFrame
```

これが有効になるのに数分かかることがあります。

- マネージャがより長い下りスロットフレームを使用していることを確認します。

```
> show status

S/N: 00-17-0D-00-00-38-0C-8C
MAC: 00-17-0D-00-00-38-0C-8C
IP6: FE:80:00:00:00:00:00:00:17:0D:00:00:38:0C:8C
HW ver: 1.2
NetworkID: 293
Base bandwidth (min links): 9000 (1)
Frame size Up/Down 277/1052. Number of working timeslots 256.
Available channels 15.
Base timeslot(TS0) 20
Network mode is Steady State
Downstream frame 1052 timeslots
Optimization is On
Advertisement is On
AP output CTS is Ready
Backbone is off
Location is off
API is Not Connected
License 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
Network Regular Mode
Total saved links 3
```

マネージャの CLI コマンドでは、より形式ばった用語である「slotframe」の代わりに、「frame」がよく使われます。802.15.4 パケットを示すのに「Frame」が使用されることはありません。

- ここで、単一モートの電源を入れ直し、そのモートが **Negotiating1** から **Operational** に遷移するのにかかる時間に注目します。この時間は、以前に確認した「高速」参加時間の約 4 倍になるはずです。

経路安定性が悪いと再試行が増すことを意味するので、経路安定性も参加時間に影響します。メッシュ・アーキテクチャがあると、いかなる個々の経路の影響も最小限に抑えられることになります。

4.2.4 ネットワーク形成と単一モートの参加

同時に参加するモートの数もネットワーク形成時間に影響します。これらのモートは限られた参加リンクおよび下り帯域幅を求めて競合する必要があるからです。この影響を調べるには、以下のようにします（ここでは、前述したより長い下りフレームを使用していることが前提です）。

1. （前述したように）ジョイン・デューティ・サイクルが 100%となるようにすべてのモートを構成します。ここでは平均電流は関係ありません。
2. 単一モートをリセットして、参加させます。これを 10 回繰り返して平均参加時間を算出します。
3. すべてのモートをリセットし、ネットワークが完全に形成されるのに要する時間、つまり、参加トレースがオンのとき、すべてのモートが **Operational** に遷移するのに要する時間に注目します。この実験を数回繰り返して、ばらつきの感触をつかむことが必要な場合があります。

5 つのモートがあると、平均的に、単一モート・ネットワーク内で 1 つのモートが通知を受信するより早く通知を受信するモートが存在するので、5 つのモートの場合の方が最初のモートの参加を認識するのが早くなることがあります。ただし、同時に同期するモートの数が多すぎると、同じ共有アクセス・ポイント（AP）リンクへの競合が生じるので、ネットワーク全体の参加時間が長くなる可能性があります。

マネージャを「高速」の下りフレームに戻してから先へ進みます。

```
> set config dnfr_mult 1
```

- マネージャをリセットし、再ログインします。

4.2.5 離脱したモートからの回復時間の測定

メッシュが重要な理由の 1 つは、経路障害に対する耐性です。スター構造やツリー構造では、1 つの RF 経路が 1 つまたは多くのデバイス・データの単一障害点を意味することがあります。メッシュを評価する場合、多くの観測者は経路障害からのメッシュの回復を調べたいと考えます。最も信頼できる方法は、デバイスの電源を切って経路障害を発生させる方法です。また、多くの観測者はデバイスの離脱後からの回復時間にも大いに関心があるので、このテストは両方に対応できます。

まず、「回復」の意味を決める必要があります。10 デバイスまでメッシュを広げ、1 つのデバイスの電源を切断したときに、電源を切断された 1 ノードだけがデータの送信を停止し、これがデータ配信時の唯一の変化である場合、ネットワークが回復したとみなします。該当ノード以外のすべてのノードが設定レートで間断なくデータを送信し続けた場合は、回復時間をゼロとみなします。ネットワークは、ネットワークの残りの部分に混乱を与えることなくノードの離脱を切り抜けたからです。もう 1 つの調査方法は、データ待ち時間のように、サービス品質（QoS）の一定の測定基準を確立することです。ノードの電源を切る前はその QoS 測定基準を満たし、次にその QoS 測定基準が低下する原因となるノードの離脱を探し、電源の再投入後 QoS 測定基準が以前の状態に戻ることを確認します。この方法では観測者からの判定の呼び出しが増えますが、報告率に依存することが厳しい時間値を割り当てるのは容易でない可能性があります。回復を識別する第 3 の方法は、1 つ以上のノードの親であるノードの電源を切断し、その親ノードの離脱をネットワークが検出して新しい親ノードを確立するのに要する時間を測定する方法です。

3 つの異なる「回復時間」テストの目的をまとめると、以下のようになります。

1. 絶え間ないデータ配信。ある 1 つのモートの電源を切断してもそれ以外のモートからのデータ配信が中断しない場合、回復時間は 0 です。データ配信が中断した場合、回復時間はすべてのノードからのデータ配信が回復するまでの時間です。
2. QoS データ配信。データ分析により、ネットワークの QoS のベースラインを設定します。ある 1 つのノードの電源を切断し、一部またはすべてのノードについて QoS 測定基準の低下に注目します。回復時間は、以前の QoS 測定基準を再度満たすまでの時間です。
3. メッシュの修復時間。メッシュを調べます。あるノードの電源を切断し、それによって他のノードの親が 1 つだけになるようにします。回復時間は、ネットワークが離脱を検出してから完全なメッシュを再確立するまでに要する時間です。

テスト 1 と 2 は基本的には同じです。ネットワークを導入して（以下のヒントを参照）、CLI に接続し、`trace station` を実行します。マネージャが受信した各パケットは、モート ID とそのパケットの待ち時間を出力します。このテキストを一定時間取り込み、その後モートの電源を切断します。電源を切断するモートはランダムに選択できますが、メッシュ内に多くの子モートがあるモートを明確に特定することもできます。ノードの電源を切断したとき、取り込みテキスト内にあるタイムスタンプの概略値をメモします。ネットワークを引き続き数分間動作させます。そうすると、この取り込みテキストからのデータをプロットして、電源が切断されたモートから最後のパケットを受信した瞬間を特定できるので、それ以降は他のノードからのデータ配信時に隔たり（差）や遅延があるかどうかを特定できます。自身で構文解析ツールとデータ分析ツールを用意する必要があります（MATLAB、Excel、Python はすべて適しています）。

最初の数モートが参加するのを待ってから残りのモートの電源を入れることにより、すべてのモートに同時に電源を入れる場合よりメッシュを迅速に形成できます。モートは、モートの参加時に通知を受信したモートを報告します。すべてのモートの電源を同時に入れた場合、AP だけが通知を送信するので、有望なモートだけが AP を可能性のある隣接モートとして報告します。モートは他の隣接モートの検出と報告を低速化してエネルギーを最小限に抑えます。参加後にマネージャがネットワークの「メッシュ化」を開始するのに最大で 5 分かかります。実際の配置では、隣接モートを検出するための遅延は重要ではありませんが、デモや評価のシナリオでは、数モートの参加を待つと直ちにメッシュ化することができます。

テスト 3 では、マネージャに別のトレース機能が必要です。初期ネットワークを配置し、メッシュが確立されていることを確認します（つまり、numparents=2 の場合は、すべてのモートに 2 つの親が存在します。ただし、ある 1 つのモートの親は厳密に 1 つだけです）。マネージャのテキスト取り込みを開始します（`trace link on`）。モートを 1 つ選択して、電源を切ります。モートの電源を切った瞬間のタイムスタンプの概略値をメモして、障害が発生した経路の検出に関連したイベントと、新しい経路の確立に関連した動作に注意します。数分後にトレースを停止し、メッシュが完全に復元されていることを確認します。「回復時間」は、モートの電源を切ったときから、取り込みテキストに記録された最終リンク追加イベントのタイムスタンプまでの時間です。予想される結果は以下のとおりです。

1. ネットワークが着手に適したメッシュだった場合は、1 つのデバイスの取り外しが原因でデータが消失したりモートが新たに離脱したりすることはないと予想されます。
2. 中程度の報告率（5 秒以下につき 1 パケット）では、QoS に障害はないと予想されます。報告率が高い場合は、1 つのデバイスの離脱に関連した待ち時間の遅延が見受けられることがあります。待ち時間が約 200 ミリ秒のモートでは、少しの間、待ち時間が約 500 ミリ秒になることがあります。回復時間は 1～2 分になります。
3. モートの電源切断は 1～2 分の間に検出されて修復されます。

評価キットに標準的にインストールされている Stargazer GUI ツールにより、メッシュの形成を確認できます。

4.3 待ち時間

4.3.1 待ち時間の測定

マネージャ CLI 統計情報トレース機能を作動させると、マネージャ側で受信した上りパケットごとに待ち時間の測定が行われます。

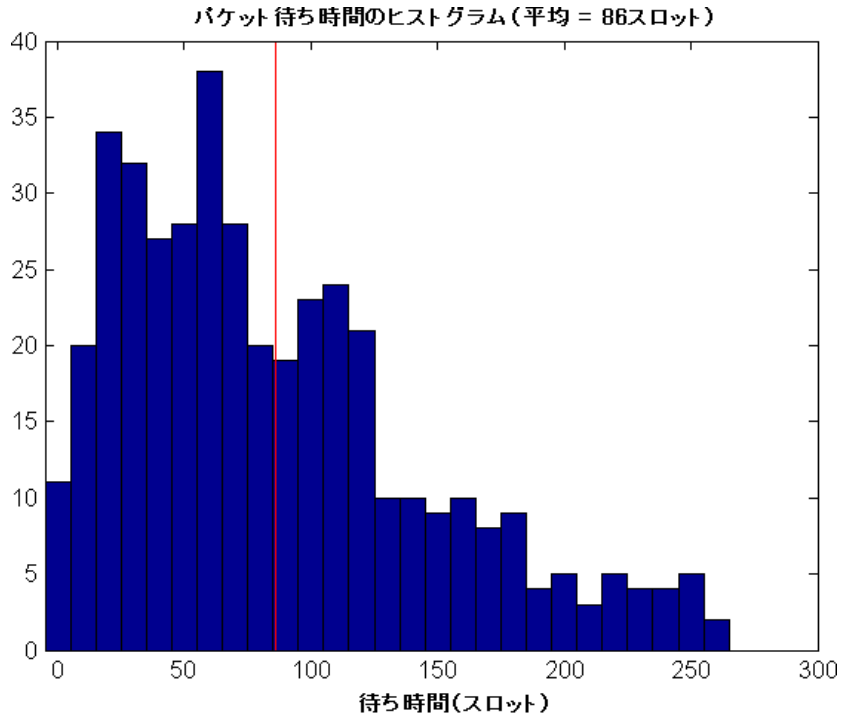
```
> trace stats on

281098 : STAT: #2: ASN Rx/Tx = 38694/38570, latency = 899, hops = 1
281100 : STAT: new average hops10=10 latency/10=56
```

このトレースは、上りパケットごとに 2 行のレポートになっています。1 行目は、パケットをマネージャ側で受信した時刻（Rx）とパケットをモート側で発生した時刻（Tx）を絶対スロット番号（ASN）で示しています。ASN は、ネットワークが起動してからか、マネージャの UTC 時刻を設定してある場合は 2002 年 7 月 2 日 20:00:00 UTC 以降に経過したタイムスロットの数です。これら 2 つの数の差に 7.25 ミリ秒/スロットを掛けると、ここで報告されている 899 ミリ秒という値になります。キューに入る遅延が存在する場合があるので、データを生成するセンサーとマネージャ側での通知との間の時間は若干長くなります。また、1 行目はこのパケットの上流に何ホップあるかも示しています。

2 行目は、マネージャがこのパケットを受信した後に統計情報を平均化していることを示しています。示している内容は、このモートの平均ホップ数（0.1 ホップ単位）と平均待ち時間（10 ミリ秒単位）です。したがって、この場合の平均待ち時間は 56*10 ミリ秒 = 560 ミリ秒です。

大量の待ち時間トレース出力結果を集めて分布図をプロットすることができます。分布図はこのようになります。



経路安定性が非常に低いわけではない限り、平均すると、1 ホップ当たりの上り待ち時間は上りスロットフレームの長さの 1/2 より短くなると予想されます。

4.3.2 スター型トポロジとの比較

ZigBee ネットワークでは、センサー・ノードの通常の報告先は電源の入った常時オンのルータです。そのため、リンクに障害が発生するまでは、データ待ち時間が非常に短くて済みます。

Dust のネットワークは、短いデータ待ち時間を実現するいくつかの手段を備えています。それは、ネットワーク全体に影響する通電バックボーン（下記の注を参照）、ネットワーク全体の基本帯域幅設定、または 1 モート当たりのサービスです。

以下の実験を試してください。

- モートを互いに数メートル以内に配置し、メッシュ型ネットワーク（デフォルト）を形成します。
  - 前のセクションで説明したように、平均待ち時間を測定します。デフォルトでは、各モートが温度データを 30 秒に 1 回送信するので、10 分より長くデータをとる必要があります。

- モートを強制的に非ルーティングにすることにより、スター型ネットワークを形成します。これは次のようにモート CLI を使用して実行します。

```
> mset rtmode 1
> mset maxStCur 20
```

- **rtmode** と **maxStCur** は不揮発性です。リセットしても電源を入れ直しても機能は持続しますが、モートがネットワークに参加する前に変更する必要があります。参加後に設定を変更した場合は、モートをネットワークに再参加させる必要があります。DC9006 ボードに接続し、CLI を使用してルーティング・モードを変更し、その後 DC9006 を切断して、別のモートでこれを繰り返す必要があります。
- ここで **maxStCur** 設定を使用して、モートが実現可能な最も消費電力が低いモードに確実に入るようにします。
- 平均待ち時間を測定します。この時点ではモートの親は AP だけなので、平均待ち時間はわずかに減少しているはずです。このスター構成では、モートがリンク障害に対して脆弱になり、待ち時間があまり改善されないので、待ち時間を改善する正しい解決策とは言えません。

- モートの設定をルーティング対応に戻し、上流のバックボーンをオンにします。これはマネージャの CLI コマンド `set config` で行います。バックボーンは、（他のフレームで使用される通常の専用スロットとは対照的に）アクセス競合スロットを持つ短いスロットフレームで、これを使用すると待ち時間の短い、多数のモートが共有する帯域幅を実現できます。実現可能な最も消費電力が低いモート構成を得るため、**nummlinks** を 0 に設定して、マネージャの下りマルチキャスト・リンクも除去します。この変更を行う前にスーパーユーザ・パスワードを入力する必要があることに注意してください。

```
> set config bbsize 1
> set config bbmode 1
> su becareful
> seti ini nummlinks 0
```

- この後、バックボーンが作動し始めるには、マネージャ・リセットが必要です。バックボーン・モードは不揮発性で、リセット中も保持されます。
- マネージャに再接続してすべてのモートを参加させたら、マネージャの CLI 待ち時間トレースをオンに戻します。

```
> trace stats on
```

- 平均待ち時間を測定します。モートはピアを介してときどきルーティングするにもかかわらず、待ち時間は劇的に短くなります。

- バックボーンをオフにします。

```
> set config bbmode 0
```

- マネージャをリセットし、統計情報トレースをオンに戻します。

- ネットワーク内の基本帯域幅を広げます。デフォルトは 9000 ミリ秒（つまり、9000 ミリ秒につき 1 パケット、または.11 パケット/秒）です。1000 ミリ秒に設定します。これも次の `set config` コマンドを使用して実行します。

```
> set config basebw 1000
```

- この後は、マネージャのリセットが必要です。
- マネージャに再接続してすべてのモートを参加させたら、CLI 待ち時間トレースをオンに戻します。
- 平均待ち時間を測定します。値は低下しているはずです。モートの発行頻度は高くなっていませんが、より多くのリンクを受け取ってきたので、待ち時間は減少します。

- 基本帯域幅の設定値を 9000 ミリ秒に戻します。

```
> set config basebw 9000
```

- 1 つのモートでサービスを要求します。
- 平均待ち時間を測定します。サービスを持つモートでは待ち時間が短くなり、そのモートが親モートになっている別のモートでは待ち時間が少し短くなる場合があることが分ります。

⚠ デフォルトでは、モートの送信リンクは上流の通電バックボーンにあるリンクだけです。これがモートの消費電力に及ぼす影響はごくわずかです。パケットを受信(してバックボーン上の他のモートにパケットを転送)するには、対象のモートで **setParameter<pwrSource>** API を呼び出す必要があります。これについては、「アプリケーション・ノート:組込みマネージャの通電バックボーンを使用した待ち時間の改善」で説明しています。これにより、ネットワーク全体の待ち時間を(上り方向または双方向で)改善できますが、モートの電流が大幅に増加します。ただし、双方向のバックボーンをオンにした場合でも、Dust のモートは標準的な ZigBee ルータよりも電流が 10 倍少なくなります。

4.3.3 消費電力と待ち時間のトレードオフ

一般に、消費電力と待ち時間はトレードオフの関係にあります。スロットフレーム全体を通じてリンクの間隔を比較的均等にする傾向があるので、モートのリンク数が多ければ多いほど、どのパケットの待ち時間も短くなります。これは、モートが同じ割合でパケットを生成する場合でも、トラフィックを転送するモートの方が転送しないモートより待ち時間が短いという意味です。また、平均待ち時間を目標にしてサービスを要求することにより、データ生成速度が低い場合であっても、モートが待ち時間を改善できることも意味します。

電力セクションでの一部の電力測定実験では、消費電力と待ち時間との間に相関があることを示しています。

4.3.4 往復の待ち時間

Dust のネットワークは、主にデータ収集ネットワークとして機能するよう設計されているので、ほとんどの帯域幅は上り方向のトラフィックで費やされます。サービスを使用することで上り方向の占める割合を改善できますが、高速(メッセージ当たり 2 秒未満)の呼び出し応答が必要な場合は、双方向のバックボーンを使用できます。以下の実験を試してください。

- マネージャ CLI を介して `io` トレースと `iodata` トレースをオンにします。
- マネージャ・モードで APIExplorer を使用して、アクノリッジされたアプリケーション・パケットをモートへ送信します。これは **sendData** API と OAP プロトコルを使用して実行します。送信元/宛先ポート = `61625` (0xF0B9)、ペイロード = `050002FF020302000101` に設定します(最後のバイトを **00** に変更すると、オフに戻すことができます)。これにより、モートのインジケータ LED が点灯し、LED が設定されたことをモートが認識します。マネージャ CLI に次のように表示されます。

```
442887 : TX ie=32:2c mesh=1222 ttl=127 asn=59614 gr=2 dst=2 rt=r4 r=1:2 tr=u:req:8; sec=s nc=0
ie=fe:00 IP=7d77 UDP=f7 sP=f0b9 dP=f0b9;

443503 : TxDone

444269 : RX ie=32:1c mesh=4002 ttl=127 asn=59699 gr=1 src=2 rt=g tr=u:req:0; sec=s nc=11
ie=fe:00 IP=7d77 UDP=f7 sP=f0b9 dP=f0b9;
```

ペイロード内の 2 番目のバイトはシーケンス番号です。複数の OAP コマンドを発行する場合、SmartMesh IP Tools Guide のオンチップ・アプリケーション・プロトコルのセクションの説明に従って、適切にこのシーケンス番号をインクリメントする必要があります。

> ⚠ 一部のトレースは、デフォルトの CLI ボーレート 9600 では処理できない量の出力を生成する場合があります。このようなトレースを有効にするには、はじめに高速 CLI 出力を有効にします。
>
> > su becareful
>
> > debug hscli on
>
> このコマンドが実行するのは CLI 速度の変更ではなく、トレース抑制の有効化だけである点に注意してください。このコマンドは小規模ネットワークでのテストのみに使用してください。通常のネットワークで使用するとマネージャの性能に影響する可能性があります。これらのコマンドは、リセットした後や電源を入れ直した後には維持されません。

これを数回繰り返し、往復の待ち時間（TX の時刻と RX の時刻との差、ここでは 1382 ミリ秒）を測定します。

- 双方向のバックボーンをオンにします。

```
> set config bbsize 2

> set config bbmode 2
```

この後、バックボーンが作動し始めるには、マネージャ・リセットが必要です。

- マネージャに再接続してすべてのモートを参加させたら、CLI 待ち時間トレースをオンに戻します。

```
> trace stats on
```

- 手順 1 の **sendData** テストを繰り返します。往復待ち時間が劇的に改善されていることが分ります。

> ℹ 通電バックボーンは、非アクティブ状態になってネットワークが再度リセットされるまではリセットしても電源を入れ直しても状態を持続します。この結果、モートの平均消費電流は 1～2mA になります。

4.4 チャンネル・ホッピングと範囲

他の大半の 15.4 ベースのシステムとは異なり、Dust の製品は Time Slotted Channel Hopping MAC を採用しています。これは、伝送が 2.4GHz 帯のすべてのチャンネル（または一部のチャンネル：下記の「ブラックリスト登録」参照）にわたって広がっているという意味です。これには、システムの性能が 1 つのチャンネルの安定性ではなく、平均的なチャンネル安定性に依存するという利点があります。以下のテストでは、1 つのチャンネルの安定性を測定する方法を示し、チャンネル・ホッピングの影響を調べます。このテストでは、802.11g Wi-Fi ルータにアクセスできることと、そのルータを特定のチャンネルに設定できることが必要です。ルータは（モート・チャンネル 4～7 をカバーする）802.11 チャンネル 6 に設定します。また、無線テストに備えてマネージャとモートを構成する必要もあります。

以下の各種 API では、チャンネルに 0～15 の番号が付けられています。これらは IEEE チャンネル番号付与方式でのチャンネル 11～26 に対応します。

4.4.1 単一チャンネル測定での無線テスト（radiotest）の使用

この説明では、マネージャの役割をトランスミッタ、モートの役割をレシーバーと位置付けていますが、役割を交換することでテストを簡単に実行できます。

- マネージャの CLI で、ログインしてマネージャを無線テスト・モードにします。

```
> login user
> radiotest on
> reset system
System reset by CLI
SmartMesh IP Manager ver 1.1.0.30.
   658 : **** AP connected. Network started
```

- マネージャがリセットしたら、マネージャがトランスミッタになるよう構成し、チャンネル 0 において、電力 +8dBm、200 パケット、80 バイトのペイロード、5 ミリ秒のパケット間遅延の条件でパケット・テストをマネージャに設定し、モートをレシーバーとして構成するまで待ってから Return キーを押します。無線テスト・モードではログインは任意です。

```
> radiotest tx pk 0x0001 8 200 80 5000
```

- モートの CLI で、無線テスト・モードに入り、リセットします。その後、チャンネル 0 で 2 分間受信するようモートを構成します。

```
> radiotest on
> reset
SmartMesh IP mote, version 1.1.0.41 (0x0)


radiotest rx 0x0001 120
```

- この時点で、マネージャの Return キーを押し、テストを開始します。マネージャが送信を終了したら、受信したパケット数をモートの CLI で記録します。

```
> radiotest stat
Radio Test Statistics
OkCnt : 998
FailCnt : 2
```

受信パケット数と送信パケット数の比は、このチャンネルの経路安定性（またはパケット成功比）です。1000 - **OkCnt** が **FailCnt** に等しくならないことがありますが、これはマネージャが自動追跡可能なパケットだけを数えるからです。

すべてのチャンネルについて繰り返します。Wi-Fi ルータが使用しているチャンネルの近辺に経路安定性の落ち込みがあることが分ります。この落ち込みが、予想した大きさとどの程度異なるか注意してください。ルータにもデューティ・サイクルがあるのでこれは部分的であり、わずかな時間にすぎません。

4.4.2 範囲テスト

radiotest コマンドは範囲テストに使用することもできます。範囲とは 2 つのデバイスが確実にパケットを交換できる距離のことですが、あいまいなものでもあります。2 つのデバイスがある距離を置いて通信する能力には、以下のすべての条件が影響するからです。

- 反射物 - 無線信号が跳ね返るもの。地表を含む
- 障害物 - 無線信号が通り抜ける必要があるもの
- 干渉物の存在
- アンテナのデザイン
- 無線電力の設定値

最も簡単なテストは、前述した単一チャンネル測定を繰り返す方法ですが、実際のネットワークではすべてのチャンネルを使用するので、それをすべてのチャンネルに対して繰り返します。パケット・エラー率は、見通し経路と跳ね返り経路の間の自己干渉により、地表からの無線の高さに大きく影響されることが分ります。モートを標高 1m に置いた場合、間隔が 50m に近づくとパケット成功率が低下することが分るので、改善するにはより高い位置に置くしかありません。何もない空き地の数メートル上空に置かれたモートは、数百メートルの距離で通信できるはずです。

このテストの終了後は、マネージャとモートを通常動作に戻します。

```
> radiotest off
OK
> reset
```

⚠ DC9003 ボードには、ダイポール・アンテナと比べて利得が極めて低い（+2dBi に対して平均-2.3dBi）組込みのチップ・アンテナが付属しています。範囲の測定では、この差を考慮に入れる必要があります。

4.4.3 ブラックリスト登録

マネージャは、既知の干渉物が存在する場合、特定のチャンネルをブラックリストに登録する API を備えています。通常は、（極めて繁忙な WiFi ルータなどの）強い干渉物があることが分っていない限り、ブラックリスト登録はしないでください。

- 5 モートのネットワークを形成します。すべてのモートが参加したら、統計情報を消去します。

```
> exec clearstat
```

- 1 時間動作させます。マネージャ CLI を使用して、ネットワーク全体としての経路安定性を調べます。干渉物によるパケットの消失が発生することはなくなります。

```
> show stat

Manager Statistics ------------------------------

   established connections : 1

   dropped connections     : 0

   transmit OK             : 1907

   transmit error          : 0

   transmit repeat         : 0

   receive OK              : 35

   receive error           : 0

   acknowledge delay avrg  : 15 msec

   acknowledge delay max   : 35 msec

Network Statistics ------------------------------

   reliability: 100% (Arrived/Lost: 1931/0)

   stability: 98% (Transmit/Fails: 983/25)

   latency: 200 msec

Motes Statistics ----------------------------------

   Mote    Received    Lost    Reliability    Latency   Hops

   #2      1019        0       100%           120       1.0

   #3      19          0       100%           1050      1.0
```

- マネージャをリセットし、チャンネル 4～7 をブラックリストに登録して、マネージャをリセットします。ブラックリストの機能はリセットしても電源を入れ直しても持続します。

```
> login user
> set config channellist 00007f0f
> reset system
System reset by CLI
SmartMesh IP Manager ver 1.1.0.30.
   658 : **** AP connected. Network started
```

- ネットワークが形成されたら、統計情報を消去し、もう 1 時間動作させて、経路安定性を確認します。

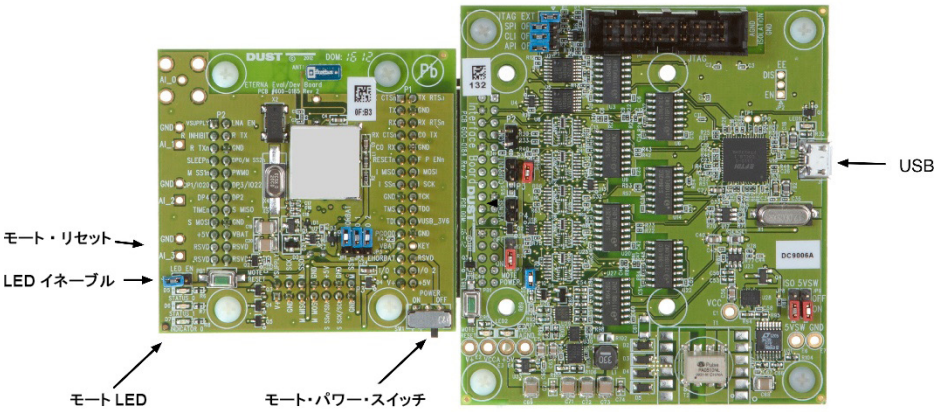
マネージャを 15 チャンネルに戻してから先に進みます。

```
> login user
> set config channellist 00007fff
> reset system
System reset by CLI
SmartMesh IP Manager ver 1.1.0.30.
   658 : **** AP connected. Network started
```

4.5 電源

このセクションでは、評価／開発ボードで電力を測定する方法について詳しく説明します。モートの電流を測定するには、DC9003 A-B/DC9006 の組み合わせを使用します。DC9006 の VSENSE と記されたジャンパの両端にオシロスコープを接続して、10Ωの直列抵抗両端の電圧降下を測定するか、CURRENT ヘッダの裏側にある赤のジャンパ(JP5)を取り外して、VSENSE ピン間に平均電流メータを接続します。

⚠ DC9003A-B ボードの LED_EN ジャンパは使用不可にしておく必要があります。そうしないと、電流測定が不正確になります。



推奨の測定方法を以下に示します。

⚠ **スレーブ・モード**になっているモートは、アクノリッジが返され 500μA を超える電流が流れるまで、起動イベントを連続して生成します。アイドル電流を正しく測定するには、SmartMesh SDK から APIExplorer に接続してこのイベントをアクノリッジする必要があります。

- アイドル電流 - モートはリセットされているが、join API コマンドは発行されていない場合。モートは**スレーブ・モード**に構成して自動参加を防止する必要があります。アイドル電流の測定後は、モートを**マスタ・モード**に戻します。
- 探索 - デューティ・サイクル 5%のときとデューティ・サイクル 100%のときに測定します。電流が約 250μA 平均から 5mA 平均に変わることが分ります。
- 参加モート - **dnfr_mult** を 4 に、**setDnframe** を fast に設定して、マネージャをリセットしてからモートを参加させます。1 時間後、マネージャはより長い下りフレームに遷移するので、平均電流を比較できるようになります。遷移前は約 43μA で遷移後は 30μA になります。CLI で「exec setAdv off」コマンドを使用して通知を停止し、平均電流が 14μA まで減少するのを確認します。
- バックボーン - bbmode を 2 に、bbsize を 2 に設定して、マネージャをリセットします。平均電流を測定します。約 500μA になります。

Dust は SmartMesh Power and Performance Estimator を用意しており、様々な使用事例でモートの平均電流を推定するのに役立ちます。テスト結果をスプレッドシートの結果と比較します。

4.6 メッシュの動作

Dust のネットワークは、自動的にメッシュを形成します。各モートには複数の隣接モートが存在し、それらを介してデータを送信または転送できます。この設定はマネージャのポリシーであり、マネージャ CLI コマンド>show config（上記参照）によって表示できます。このポリシーを設定する config 要素は numparents であり、上記から分るように、デフォルト値は 2 です。マネージャは、メッシュ内にループがないことを前提として、すべてのモートの親の数が、numparents で示された親の数以上であることを確認します。この略図をホワイトボードで作成してみてください。1 モートのネットワークでは、その 1 モートの親は AP モートだけになることがわかるでしょう。2 つ目のモートを追加すると一方のモートの親は 2 つになりますが、ループの発生を防ぐため、もう一方のモートの親は 1 つのままです。追加するモートの数に関係なく、「片親のモート」が常に 1 つ存在します。

4.6.1 メッシュのテスト

Stargazer GUI アプリケーションを使用すると、形成したときのメッシュを確認できます。まだインストールしていない場合は、今すぐインストールしてください。また、このアプリケーションを使用するには、シリアル・マルチプレクサを設置することも必要です。

親が 2 つのモートは、ほとんどの動作条件で堅牢性と電力消費のバランスが最適です。堅牢性は複数の親を持つことで実現されます。1 つまたは多数のモートの親であるモートをネットワークで見つけることを推奨しますが、その確認方法は次のとおりです。該当モートの電源を切ります。該当モートからのデータ配信は停止しますが、子モートは引き続きデータを送信します。前述したように「trace stats on」を使用して、すべての子モートからのライブのデータ・ストリームを示します。これらのデータは子モートの他の親を通じて送信しているデータです。数分の間、待ち時間が一時的に長くなる可能性があります。また、このトレースをファイルに取り込むと、トレースを図示できます。マネージャは、システム内の親とリンクを再度割り当てて、電源を切ったモートの離脱を補うのに数分かかります。

4.6.2 メッシュ・ポリシーの変更

パラメータ「numparents」には、4 つの値（1、2、3、または 4）のいずれかを指定できます。numparents に 1 を設定すると、ネットワークはメッシュ型ではなくツリー型になります。各モートの親は 1 つになり、その 1 つの親は AP ではないことがあります。ツリー型はマルチホッピングをサポートするという点でスター型より優れています。しかし、スター型と同様、ツリー型ネットワークには非常に多くの「単一障害点」が存在します。ツリー内のある 1 つのモートの親の電源を切ると、子モートは他のどのデバイスとも接続していないので、ネットワークから脱落します。この子モートはリセットし、ネットワークに再参加する必要があるので、データを失う可能性が高くなります。ツリー構造の消費電力はわずかに低くすることができます。それは、各デバイスが受信する必要があるのは 1 つの親からの下り方向のメッセージだけであり、維持しなければならないのは 1 つの親との同期だけだからです。それに加えて、個々のパケットはメッシュの周囲を「徘徊する」ことができないので、データ待ち時間はツリーにより依存すると考えられます。当社では、ネットワークの崩壊やデータ消失の可能性の方が消費電力と待ち時間の潜在的な利点を大幅に上回っていると考えています。

2 はデフォルト値であり、堅牢性と電力のバランスが取れています。numparents を 3 または 4 に設定すると、システムの「メッシュ性」が増し、ネットワークの堅牢性が高まりますが、いくぶん消費電力が高くなるという代償を払うことになります。経路が頻繁に遮断されることが分っているネットワークでは、親の数を増やすことを推奨します。アプリ

ケーション・ノート「干渉の影響の特定と軽減」を参照してください。numparents = 4 に設定することで、いくつかのノードの適度な移動性がうまくサポートされます。

4.7 データ・レート

SmartMesh SDK には、次の 2 つのサンプル・アプリケーションが収録されています。1 つは TempMonitor で、SmartMesh IP ネットワーク内の 1 つ以上のモートで温度公開のレートを設定できます。もう 1 つは PkGen で、特定のサイズのパケットを特定のレートで生成できます。これら 2 つのアプリケーションを同時に使用して、温度データとシミュレーション・データの両方を異なるレートで同時に生成することができます。また、Stargazer アプリケーションを使用して、温度レポートだけでなくアナログ・データやデジタル・データを構成することもできます。

これらのアプリケーションにより、以下の質問に答えることができます。

- ネットワーク内でのデータの実際の流れはどう見えますか？
- 単一モートはどのくらい速く移動できますか？
- すべてのモートはどのくらい速く移動できますか？上流のバックボーンがオンのときはどうですか?

最終的なデータ・レートは、モートの数、各モートが要求するサービス（またはサービスに加えて基本帯域幅あるいはサービスの代わりに基本帯域幅）、トポロジ、および経路安定性により異なりますが、このツールにより、ある種の柔軟性の好例を期待できます。

5 アプリケーション・ノート：組込みマネージャのサブスクライブ API フィルタの使用法

5.1 サブスクライブ・フィルタ

subscribe API は、マネージャがどの通知をホスト・アプリケーションに送信するかを構成するために使用します。通知のリストは、以下の 2 つのビットマップのいずれかで指定します。

- **filter** ビットマップは、送信対象の通知を指定します。
- **unackFilter** ビットマップは、非アクノリッジ制御（ベストエフォート）通信を使用して送信する通知を指定します。マネージャは、ホストの動作に関係なく、生成された時点で各通知を送信します。デフォルトの動作は、アクノリッジ制御通信を使用して送信する方式で、ホストが各通知にアクノリッジを返すのを待つ間、後続の通知パケットはキューに入ります。

unackFilter ビットマップのビットは、対応するビットが **filter** ビットマップに設定されていない限り意味はありません。

5.2 アクノリッジ制御か非アクノリッジ制御か？

リンクに誤りがある可能性が低い場合は、ホスト UART が常に使用可能な状態であり、十分なバッファがあります（例えば、ホストが PC である場合）。あるいは、アプリケーションがある程度の量のデータ消失を許容できる（例えば、レポートがなくても警戒状態にならない）ので、非アクノリッジ制御通信を **data** 通知および **ipData** 通知に使用できます。その他の通知は頻度が非常に低く、欠落しないことがより重要なので、常にアクノリッジ制御通信を使用する必要があります。非アクノリッジ制御通信はアクノリッジ制御通信より高速です。パケット間の遅延を最小で 1 ビット時間（約 10 µs）にすることが可能であり、アクノリッジの送信に時間がかからないからです。現在のシリアル・ポート速度（115200bps）では、非アクノリッジ制御通信を使用してマネージャのデータ・スループットを最大限に高めることが必要な場合があり、パケット生成速度が 10 パケット/秒より速い場合は非アクノリッジ制御通信の使用を検討する必要があります。

アクノリッジ制御の伝送方式により、ホストは以下のような多少のエラーをものともせず、各パケットを確実に受信します。

- フレーミング・エラー（通常はパケット送信開始時にホストがスリープ状態の場合に発生）
- ビット・エラー（SNR が不十分なリンク（例：遮蔽が不十分なケーブル）によって発生）
- ホストに Rx バッファがないか、別の理由でホストの受信準備が完了していない

マネージャは 200 ミリ秒待ってから各パケット受信の再試行を行います。パケット受信が 3 回再試行されてもアクノリッジが返されない場合、マネージャはそのセッションが途切れたとみなし、保留中のパケット（とキューに入っているその他の通知）を破棄して通信を切断します。SmartMesh IP Embedded Manager API Guide に説明されているように、ホストはセッションを再確立して、通知を再サブスクライブする必要があります。

マネージャは少数のパケットをキューに入れてから、AP からのパケットを拒否します。生成率と経路安定性によっては、モートのキューがいっぱいになる前に、ネットワークが各パケットの最大再試行回数を許容できない場合があり、モートがパケットを拒否し始めます。これにより、パケットの消失や陳腐化が発生することがあります。全体のパケット率が 1 パケット/秒より低い場合、通知の再試行が原因でネットワークがパケットを廃棄することはありません。

6 アプリケーション・ノート：組込みマネージャでの SmartMesh IP ネットワークの健全性のモニタ

組込みマネージャの IP ネットワークは、ネットワークを管理して最適化するのに必要な最小限の統計情報を維持します。これらのネットワークでは、履歴による傾向を追跡する目的では統計情報を集計しないので、モートごとの細分性が得られることは多くありません。ただし、この統計情報の多くの情報源（すなわち、モートの加工されてない健全性レポートおよび状態レポート）は、ネットワークの健全性情報とデバッグのフィードバックを提供するために、クライアント・アプリケーションがサブスクライブできる通知の形式で利用できます。いくつかの例を以下に示します。

- 状態の変化を監視することにより、リセットしているモートがあるかどうかをアプリケーションが通知できます。
- 隣接モートの健全性レポートを監視することにより、経路に障害が発生した場合に回復に適した十分な数の隣接モートがすべてのモートに存在することをアプリケーションが確認できます。

更に、アプリケーションはマネージャの API を使用してネットワークに関する詳細情報を入手できます。本書では、ネットワークの健全性をモニタするために、正常に機能するネットワーク上で継続的に実行できるいくつかのテストについて詳しく説明します。

この処理は 2 つの部分に分れます。1 つ目は、マネージャからの通知の出力に関してアプリケーションが継続的にモニタする必要がある一連の通知です。2 つ目は、必要な残りの情報を収集するためにアプリケーションが定期的に行う必要がある一連の API 呼び出しです。アプリケーションはネットワークと同時に起動して、ネットワークの全存続期間の健全性をモニタするのが理想です。

すべての通知データを保管することを推奨します（毎日のログに分割することになるでしょう）。記憶容量が心配な場合は、項目（例：経路 x RSSI）ごとに最大値、最小値、FIR フィルタ処理済み平均値を保管すれば、フィードバックするには十分と考えられます。

> ⚠ LTC5800/590x マネージャの場合、マネージャのリセット後に同じモート ID が割り当てられると保証されるわけではありません。ライフタイムを通じた統計情報を提供する場合、マネージャのリセット時にモート ID とモートの MAC アドレスをアプリケーションでマッチングする必要があります。

6.1 健全性レポート

ネットワークの各モートが送信する健全性レポート（HR）は 3 種類あり、それぞれ 15 分に 1 回送信されます。モニタする必要があるのは、隣接モート健全性レポートとデバイス健全性レポートのみです。3 種類目の隣接モート検出の健全性レポートが示す情報には、マネージャ API を介する方がより容易にアクセスできます。HR の通知はマネージャから非同期で到着するので、アプリケーションは HR の通知をモニタする必要があります。

6.1.1 隣接モート HR

この HR は、特定のモートが関与しているすべての使用済み経路に関する情報を示します。この HR から取り込む情報は、**neighborId**、**rssi**、**numTxPackets**、**numTxFailures** です。また、通知のタイムスタンプにも注意します。

標準的な使用では、ほとんどの経路が子から親への方向でより多く使用されますが、どちらのモートも経路に関する統計情報を報告します。経路の安定性において十分な統計的有意性を得るためには、**numTxPackets** > 10 の経路のみを対象とします。この場合、安定性をパーセントで表すと 100*(1-**numTxFailures/numTxPackets**)になります。この安定性をこの経路の **rssi** と一緒に記録する必要があります。これらの 2 項目を経路ごとに 1 つの組として記録します。

6.1.2 デバイス HR

この HR は、モートの内部動作に関する情報を示します。ここでは、アプリケーションからメッセージを受信したときにモートがどの程度正常に動作していたかを知らせる値を抜き出します。これは、後でネットワーク全体の可用性を計算するときに考慮に入れます。この HR から取り込む情報は、**numTxOk** および **numTxFail** です。

モートの可用性をパーセントで表すと、100*(1-**numTxFail**/(**numTxFail**+**numTxOk**))となります。

ネットワーク全体の可用性を計算するには、新しい変数を 2 つ定義して、その初期値をゼロにします。これらの変数を **appTxPk** および **appTxFail** と呼びます。

変数の数値を各デバイス HR と一致させ続けるには、次のようにします。

- **appTxPk += numTxOk + numTxFail**
- **appTxFail += numTxFail**

6.2 周期的な API 呼び出し

以下の API 呼び出しを使用して 15 分ごとにマネージャをポーリングし、HR の周期と一致させることを推奨します。モートごとに 0 から始めて順に **getNextPathInfo** を繰り返します。

6.2.1 getMoteConfig

Store：**macAddress**、**moteId**、**isAP**

このコマンドは、ネットワーク内に存在するすべてのモートの MAC アドレスを取得する便利な方法です。これを順に繰り返し、**next** フラグを使用して MAC アドレス／モート ID のすべての対を取得して、**macAddress**=0 から作業を開始します。このアドレス対応関係を保管すると、アプリケーションは、報告可能な残りの健全性情報を MAC アドレスまたはモート ID で解読できます。通常、この API によって返される情報が変わることはありませんが、それはネットワークに新しいモートが追加されないことが前提です。

6.2.2 getMoteInfo

Store：**numGoodNbrs**、**requestedBw**、**totalNeededBw**、**assignedBw**、**packetsReceived**、**packetsLost**

この API によって返される情報は、マネージャがモートからパケットを受信したときに、マネージャによって継続的に更新されます。可能なネットワーク接続がすべて完全に検出されると **numGoodNbrs** の値は同じ値を維持しますが、最適化により、ネットワーク全体を通じて帯域幅分布の要件が変更される可能性があります。

ネットワーク全体の信頼性を計算するには、新しい変数を 2 つ定義して、その初期値をゼロにします。これらの変数を **networkRX** および **networkLost** と呼びます。

変数の数値を各 **getMoteInfo** 呼び出しと一致させ続けるには、次のようにします。

- **networkRX** += **packetsReceived**
- **networkLost** += **packetsLost**

6.2.3 getNextPathInfo

Store：**source**、**dest**、**direction**、**numLinks**、**quality**、**rssiSrcDest**、**rssiDestSrc**

マネージャはすべての経路情報を維持しますが、粗く平均的な形式のみとなります。HR の動作を注視することにより、このすべての情報をより精細な分解能で取得できます。

6.3 テスト

以下に示すのは、収集したすべてのデータに対して実行し、ネットワークの健全性についての警告を示すことができるテストです。

> ⛔ 下記のテストで示される警告フラグを立てるべき状況は、ネットワークでの発生を**回避する**必要のある状況です。

上流のリンクの数を計算するには、各経路で報告された **direction** を調べます。

- TX リンクの合計は、**direction**=2 の **numLinks** を合計した値です。
- RX リンクの合計は、**direction**=3 の **numLinks** を合計した値です。
- 親の総数は **direction**=2 の経路の数です。

6.3.1 AP のみの場合

以下のテストは AP に対してのみ実行します。このデバイスは **isAP** フラグを設定することによって識別します。

リンクが飽和状態に近い AP

前述したように、**direction**=3 のすべての経路について **numLinks** を合計します。AP が外付け RAM なしでサポートできる RX リンクの数は 150 なので、RX リンク数が 140 を超えた場合は、どんな追加サービスも受け入れられない危険性があります。外付け RAM のある AP がサポートできる RX リンク数は 250 なので、230 という RX リンク数は、警告を出すのに適した閾値です。

6.3.2 全モートにわたる繰り返し

以下のテストは、各モートで個別に実行されます。

良質な隣接モートが 3 未満の場合

ネットワーク内のすべてのモートには、品質スコアが 0.5 を超える「良質」の隣接モートが 3 つ必要です。マネージャは良質な隣接モートの数を常に把握しているので、各モートに対して 3 つ以上の良質な隣接モートをマネージャが **numGoodNbrs** フィールドで報告していることを確認します。

リンクが飽和状態に近いモート

前述したように、すべての経路について **numLinks** を合計して、上流の TX リンク数と RX リンク数を合計します。Eterna モートが格納できるのは合計 200 リンクなので、保有リンク数が 150 を超えるモートはボトルネックが生じる危険性を示す可能性があります。

AP より多い参加回数

モートが **Oper** 状態に遷移した回数を数え、AP についても同じ回数を数えます。モートの参加回数が AP より多い場合、モートはリセットして再参加していることを意味します。モートのリセットは、特にそれが集団で発生した場合は、ネットワークに何らかの異常（干渉、モートが離れすぎている、リセットを引き起こすハードウェアの問題など）があるという最大の兆候です。

複数の片親モート

片親のモートがちょうど 1 つ存在する必要があり、このモートは親として AP を保有している必要があります。そうでない場合は、ネットワーク内のいくつかのモートに不十分な接続が存在します。他の片親のモートの親の近くに中継器モートを追加することを検討してください。

不十分な帯域幅

マネージャは要求帯域幅と割り当て帯域幅をモートごとに把握しています。ここで報告された帯域幅はモートのリンク間での平均時間なので、数値が低いと 1 秒当たりのリンク数が増えることになります。**totalNeededBw** > **assignedBW** であることを確認してください。

その他のテスト

- モートはモートのキューの占有状況に関する詳細も報告します。この付加データを使用して、より多くの健全性チェックを実行できます。
- **getMoteInfo** API は平均待ち時間に関する情報を返します。これを長時間モニタし、待ち時間が予想外に長くなった場合はフラグを立てます。

6.3.3 全経路にわたる繰り返し

この HR 通知は、ネットワーク内の各経路の 15 分間のスナップショットを取り込みます。これらのスナップショットを繰り返し処理すると、低品質の経路を特定できます。

安定性の低い RSSI

経路が以下のいずれかに該当する場合、フラグを立てます。

- RSSI が-80dBm を超え、かつ安定性が 50%未満
- RSSI が-70dBm を超え、かつ安定性が 70%未満

これらの経路は試作品の「RSSI-安定性」滝型曲線の下に位置します。ここにいくつか孤立値がある場合、あまり心配する必要はありません。モートのいずれかの閾値より低い安定した点は、ハードウェアの問題か、近辺での干渉を示している可能性があります。詳細については、アプリケーション・ノート:「干渉の影響の特定および軽減」を参照してください。

HR ごとに通知時間を取り込んだので、孤立点の時間をプリントアウトして、孤立点が一時的に集まったかどうかを確認するのは簡単です。

6.3.4 ネットワークの検査

以下はネットワーク全体として検査できる特性です。

信頼性 < 99.9%

マネージャは全体的な信頼性の粗い基準を、最も近いパーセント値に丸めて、**getNetworkInfo** API から返された **netReliability** で維持します。信頼性は「9 の桁数」という尺度で表されることが多いので、高精度な値が要求されます。SmartMesh ネットワークは、9 が 3 桁以上並ぶ信頼性を目標に設計されています。

上記の変数から計算した総合的な信頼性は、100.0*(1-**networkLost**/(**networkLost + networkRX**))です。

可用性 < 99%

ネットワークの平均可用性を得るには、以前に測定したモートごとの可用性の平均値をとります。

6.4 グラフ化

アプリケーションを継続的に実行する場合、上記の量はそれぞれ 15 分の分解能でプロットされます。就業日と夜間の間で干渉や動いている機械類の量が大きく異なる産業環境に配置されたネットワークでは、ネットワークの安定性および待ち時間に日々のリズムがあることがこれまでに分っています。これらの測定基準を長時間追跡すると、たいていの場合は 1 回のスナップショットよりも多くのことが判明します。

7 アプリケーション・ノート：SmartMesh IP のデータ発行

このノートでは、OEM マイクロプロセッサのファームウェア・アプリケーションがネットワーク内のモートに接続し、ワイヤレス・ネットワークを介してモートにデータを送信させるために必要な具体的な手順について説明します。SmartMesh IP User's Guide と SmartMesh IP Mote API Guide の主要な概念はここに集約されています。ユーザ・ガイドでは、モートと OEM マイクロプロセッサがどのようにやりとりし、一緒になってシステムを形成するかの詳細を説明しています。

7.1 要求パケットと応答パケットの形式

SmartMesh IP Mote API Guide で説明されているように、すべてのパケットは HDLC カプセル化を使用する必要があります。これは、パケットの前後にフレーミング・バイト 0x7E があることを意味します。対象のパケットについて 2 バイトのフレーム検査シーケンス（FCS）を計算し、末尾のフレーミング・バイトの前に付加する必要があります。Mote API Guide の Protocol セクションにある Packet Format で説明されているように、ペイロードまたは FCS の 0x7D および 0x7E のインスタンスにはエスケープ処理が必要です。

フラグ	ペイロード	FCS	フラグ
7E	パケット・ペイロード	2 バイト	7E

7.1.1 ヘッダ

要求パケットと応答パケットのペイロードは、どちらも 3 バイトのヘッダで始まります。先頭バイトは送信先または応答先のコマンドまたは通知のタイプを示します。次のバイトは残りのペイロードの長さです（ヘッダまたは応答コードは数えません）。3 番目のバイトは一連のフラグです。ヘッダとフラグの詳しい説明については、「Mote API Guide」の「Protocol」セクションにある「Packet Format」を参照してください。

要求

ヘッダ	ペイロード
3 バイト	要求ペイロード

応答

ヘッダ	応答コード	ペイロード
3 バイト	1 バイトの応答	応答ペイロード

⚠ 長さバイトには 3 バイトのヘッダまたは応答コードは含まれていません。

7.1.2 フラグ・バイト

3 バイトのヘッダの 3 番目のバイトはフラグ・バイトです。現在、フラグ・バイトで使用できるのは以下の 3 ビットだけです。

ビット 0 - 要求／応答:ビット 0 はパケットが要求の場合はクリア(0)され、パケットが応答の場合はセット(1)されます。アクノリッジ(ACK)パケットは応答パケットなのでこのビットをセットしておく必要があります。

ビット 1 - パケット ID:OEM マイクロプロセッサによる新しい要求パケットごとに、パケット ID ビットを切り替える必要があります。

ビット 3 - SYNC ビットの設定ルール:SYNC ビットは、OEM マイクロプロセッサがモートに送信する先頭の要求パケットでセット(1)する必要があります。後続の要求に対しては、このビットをクリア(0)する必要があります。SYNC ビットは起動イベント・パケットにセット(1)されます。このパケットは、モートから OEM マイクロプロセッサに送信された最初の要求パケットです。

7.2 基本手順

モートをネットワークに参加させ、データの発行を開始するために、OEM マイクロプロセッサが実行する必要がある基本手順は全部で 7 つあります。OEM マイクロプロセッサのシリアル・ポートのセットアップは、その後の通信を正常にするためには非常に重要な第 1 段階です。単純な発行アプリケーションのサンプル・ファームウェアを生成するために必要なのは、以下の処理をコード化し、マイクロプロセッサとモートの複合システムを構築してデータの発行を開始することだけです。手順は以下のとおりです。

1. シリアル・インターフェースをセットアップする
2. モートの起動イベントにアクノリッジを返す
3. 参加前のモートの構成を実行する
4. **join** コマンドを発行する
5. 参加の進行をモニタする
6. データを発行する
7. **boot** イベントを受信したら、3 に戻る

1. シリアル・インターフェースをセットアップする:OEM マイクロプロセッサとモートとの通信は、デフォルトでは 4 線式プロトコル(UART モード 4:TX、RX、UART_TX_CTSn、UART_TX_RTSn の各線)を使用して API シリアル・ポート上で行われます。デフォルトでは、API ポートの設定は、ボーレートが 115.2Kbps、8 データ・ビット、パリティなし、1 ストップ・ビットです。

⚠ LTC5800-IPM のボーレートは、ヒューズ・テーブルのプログラミングをモート上で更新することにより、必要に応じて 9600 に変更できます。

2. モートの起動イベントにアクノリッジを返す:モートは、電源投入(またはリセット)時に必ず起動イベント・パケットを API ポートに送信します。起動イベント・パケットの内容は次のとおりです。

```
7E 0F 09 08 00 00 00 01 01 00 00 00 00 D7 67 7E
```

モートは、OEM マイクロプロセッサによって明示的にアクノリッジが返されるまで、このパケットを送信し続けます。シリアル・ポートの設定が正しい場合、マイクロプロセッサは受信バッファにこのパケットが現れるのを認識できるはずです。マイクロプロセッサのシリアル・ポートが正しく構成されていない場合は、オシロスコープまたはロジック・アナライザを使用してこのパケットをモートの Tx ピンで観察できます。電源投入後、このパケットを受信するまで、通常は約 1 秒かかります。

> モート・ソフトウェアのバージョンによって、パケット ID ビットが 0 または 1 のどちらかになります。つまり、起動イベントのフラグ・バイトは 0x08 または 0x0A になります。

このイベントの ACK パケットの内容は次のとおりです。

```
7E 0F 00 01 00 FF 57 7E
```

> ACK のパケット ID が受信パケットのパケット ID と一致する必要があります。ここでは、0x01 または 0x03 のどちらかになります(フラグ・バイト付きのパケットの場合はそれぞれ 0x08 または 0x0A)。

OEM マイクロプロセッサは、モートによる起動イベント・パケットの送信を停止するため、このパケットに応答する必要があります。この ACK に応答すると、モートは状態 0(**Init**)から状態 1(**Idle**)に移ります。

3. 参加前のモートの構成を実行する:モートの起動イベントに対してアクノリッジが返され、モートが **Idle** 状態である場合、そのモートはネットワークへの参加準備が完了しています。**join** コマンドを発行する前に、1 つまたは複数の参加前構成設定を変更したい場合があります。これらの設定は、最初は工場出荷時の設定のままになっていることがありますが、動作が最適になるように後で調整してください。これらの構成パラメータの一部を以下に示します。

- **setParameter <networkId>**:デフォルトのネットワーク ID は 0x04CD(1229)です。
- **setParameter <joinKey>**:デフォルトの参加鍵は 0x 445553544E4554574F524B53524F434B です。最高レベルのセキュリティを確保するため、各モートには固有の参加鍵が必要です。
- **setParameter <joinDutyCycle>**:デフォルトは 0xFF つまり 100%(全部で 255)です。

「Mote API Guide」を参照してください。「Commands」セクションでは各 API の詳細を説明しており、「Definitions」セクションでは引数の符号化について説明しています。

以前は別の値に設定されていたと仮定すると、**joinDutyCycle** パラメータを 0xFF つまり 100%(255)に変更するパケットの内容は次のとおりです。

```
7E 01 02 08 06 FF 2F 60 7E
```

ヘッダ = 01 02 08 なので、これは **setParameter** コマンド(0x01)、ペイロードの長さ = 2 バイト(0x02)、およびビット 3 を設定したフラグ・バイト(SYNC、要求、パケット ID = 0)です。

ペイロード = 06 FF なので、これは変更される **joinDutyCycle**(0x06)で、値は 100%、つまり 255(0xFF)です。

⚠ ここでは、これをモートに送信する最初のコマンドとしているので、SYNC ビットは“H”です。これが最初の要求パケットではない場合、OEM マイクロプロセッサはモートに送信し、SYNC フラグの設定解除(ビット 3 = 0)を確認します。

その後、モートは次の応答内容で応答します。

```
7E 01 01 09 00 06 A0 21 7E
```

ヘッダ = 01 01 09 なので、これは **setParameter** コマンド(0x01)、ペイロードの長さ = 1 バイト(0x01)、およびビット 3 と 0 を設定したフラグ・バイト(SYNC、応答、パケット ID = 0)です。

応答コード = 00 なので、コマンドにエラーはありません(0x00)。

ペイロード = 06 なので、これは変更される **joinDutyCycle** です(0x06)。

4. join **コマンドを発行する**:これで参加前構成が完了したので、OEM マイクロプロセッサは、ネットワークへの参加処理を開始するよう促す **join** コマンドをモートに発行する準備が整いました。これがモートへの最初のパケットではないと仮定すると(この場合は SYNC=0)、参加パケットは次のようになります。

```
7E 06 00 02 07 33 7E
```

モートはアクノリッジを返して次のように応答します。

```
7E 06 00 03 00 2C 9D 7E
```

5. 参加の進行をモニタする:参加の進行中に、モートはアクノリッジが必要ないくつかのイベント通知を OEM マイクロプロセッサに送信します。これらの通知は次のとおりです。

モートの **joinStarted** 通知:

```
7E 0F 09 02 00 00 01 00 03 00 00 00 00 C6 DB 7E
```

OEM マイクロプロセッサのアクノリッジ:

```
7E 0F 00 03 00 4F 64 7E
```

モートの **operational** 通知:

```
7E 0F 09 00 00 00 00 20 05 00 00 00 00 A5 A2 7E
```

OEM マイクロプロセッサのアクノリッジ:

```
7E 0F 00 01 00 FF 57 7E
```

モートの **svcChange** 通知:

```
7E 0F 09 02 00 00 00 80 05 00 00 00 00 29 7A 7E
```

OEM マイクロプロセッサのアクノリッジ:

```
7E 0F 00 03 00 4F 64 7E
```

6. データを発行する:独自のパケットを送信するには、SmartMesh IP Mote API Guide で説明されているように、**sendTo** コマンドを使用する必要があります。モートはネットワーク・ソケット・インターフェースを使用して、マネージャまたはインターネット上の IP ホストにデータを送信します。特定の宛先にデータを送信するには、その前にソケットを開いて宛先にバインドする必要があります。パケットのデフォルトの宛先はマネージャです。この処理については、SmartMesh IP User's Guide の Communications セクションで説明されています。

手順は以下のとおりです。

1. **openSocket** コマンドを呼び出して通信ソケットを開きます。これにより、ソケットに **socketID** が割り当てられます。現時点でサポートされているのは、UDP ソケットのみです。
2. **bindSocket** コマンドを呼び出して、ソケットを **destPort** ポートにバインドします。これは、**sendTo** コマンドで使用するポートです。
3. **sendTo** コマンドを使用してデータを送信します。ペイロードおよびソケット情報以外に、**serviceType**、**priority**、**packetId** を指定する必要があります。詳細については、Mote API Guide の **sendTo** のセクションを参照してください。現在サポートされているのは(遅延ではなく)帯域幅サービスだけです。オープン・ソケットに対して **sendTo** を繰り返して呼び出すことができます。
4. この宛先にデータを送信する必要がなくなったら、**closeSocket** コマンドを呼び出します。これにより、ポートのバインドが削除され、ソケットに割り当てられたメモリが解放されます。パケットごとにソケットを閉じる必要はありません。

各手順では、マイクロプロセッサがパケットをモートに送信し、その後アクノリッジを受信することが必要になります。

1. **openSocket を呼び出す** - これにより、UDP ソケットが開きます。

```
7E 15 01 00 00 F4 0B 7E
```

モートのアクノリッジは次のとおりです。ソケット ID 22（0x16）を返します。

```
7E 15 01 01 00 16 B3 6E 7E
```

2. **bindSocket を呼び出す** - 任意のポートを使用できますが、0xF0B8～0xF0BF の範囲内のポートを使用した場合、ペイロードは最大になります。ここでは、以前に取得した UDP ソケットをポート 0xF0B8 にバインドします。どのポートを使用するかについて、モートと宛先が合意している必要があります。別のポートを使用する特別な理由がない限り、デフォルトの 0xF0B8 は適切です。

```
7E 17 03 02 16 F0 B8 D3 9B 7E
```

モートのアクノリッジは次のとおりです。

```
7E 17 00 03 00 26 42 7E
```

3. **sendTo を呼び出す** - このパケットでは、サンプル・データ 0xAABBCCDDEE が、パケット ID 0（0x0000）と共にマネージャ（**destAddr** = 0xFF0200000000000000000000000000002）に送信される**ペイロード**です。その他のフィールドについては、「SmartMesh IP Mote API Guide 」を参照してください。ペイロードが異なる場合は FCS も異なることに注意してください。

```
7E 18 1C 00 16 FF 02 00 00 00 00 00 00 00 00 00 00 00 00 00 02 F0 B8 00 00 00 00 AA BB CC DD EE
AF B2 7E
```

モートのアクノリッジは次のとおりです。

```
7E 18 00 01 00 7F C3 7E
```

パケットを Tx キューに送信すると、モートは送信が終了した時刻を示す通知を送信します。これにも同様に応答する必要があります。

パケット ID が 0（0x0000）のモート **txDone** 通知は次のとおりです。

```
7E 25 03 00 00 00 00 A4 7B 7E
```

マイクロプロセッサのアクノリッジは次のとおりです。

```
7E 25 00 01 00 02 04 7E
```

⚠ **sendTo** コマンドでパケット ID を 0xFFFF に設定すると、**txDone** 通知は生成されなくなります。

4. **closeSocket** を**呼び出す** - メッセージをこの宛先に送信するのが完了したら、使用していたソケットを閉じることを推奨します。よくあるのは、通信がインターネット・ホストからのメッセージによる応答である場合で、有限の「トランザクション」が行われている場合です。周期的な発行が設定されているセンサーでは、ソケットが閉じられるのは、マイクロプロセッサがリセットする必要がある場合だけです。

これは **closeSocket** コマンドにより、次のように行うことができます。

```
7E 16 01 02 16 3E 68 7E
```

モートのアクノリッジは次のとおりです。

```
7E 16 00 03 00 8D 5E 7E
```

7.3 次のステップ

データ・ペイロードをマネージャに正常に送信したので、「Mote API Guide」で他のコマンドやカスタマイズ・オプションの詳細を調べることができます。

- 帯域幅サービス - いったん**稼働**状態になると、モートはワイヤレス・ネットワークを介してデータを送信可能になります。デフォルトでは、すべてのモートがサービスを要求することなく、9 秒間隔（IP ネットワークの**基本帯域幅**）でデータを発行できるように IP ネットワークは構成されています。これで常に十分である場合は、OEM マイクロプロセッサがマネージャからサービスを要求する必要はありません。すべてのモートが同じレートでデータを発行する状況で、（温度、湿度、電圧、電流のような）センサー・データをアプリケーションが 9 秒より短い間隔で要求する場合、これは**同種帯域幅**ネットワークであり、基本帯域幅は広がる可能性があります。アプリケーションが、様々なデバイスに対して様々なレートでデータ発行を呼び出す場合、これは**異種帯域幅**ネットワークであり、すべてのデバイスがサービスを要求します。サービスは実装が容易で最も柔軟性が高いので、OEM インテグレータは常にサービスを使用することを推奨します。サービスについては、SmartMesh IP User's Guide の Services（サービス）セクションと、SmartMesh IP User's Guide の Bandwidth and Latency（帯域幅とレイテンシ）セクションで説明されています。

- 無線テスト・コマンド - これらのコマンドは製造検査に使用して、最上位の組み立て（例：アンテナが正しく接続されていること）を検査することができます。無線テスト・コマンド（**testRadioTxExt** および **testRadioRx**）については、SmartMesh IP Mote API Guide で説明されています。

- ソケットと UDP ポート - これらについては、SmartMesh IP User's Guide の Communication（通信）セクションで詳しく説明されています。

8 アプリケーション・ノート：組込みマネージャのネットワークでの通知の管理

WirelessHART マネージャとは異なり、SmartMesh 組込みマネージャはネットワーク状態に応じて通知を制御しません。通知は、各モートで約 14μA の平均電流を消費するので、低トラフィックのモートにとっては大きな値になります。LTC5800-IPR は **setAdvertising** API によって通知の有効化／無効化が可能なので、このオーバヘッドを容認できない場合には、ホスト・アプリケーションがロジックを実装して通知を制御できます。このアプリケーション・ノートでは、この目的で状態マシンの例を示します。

> ⚠ 通知をオフにすると、新しいモート、または**離脱**したモートは、ネットワークの検出および参加ができません。そのため、通知はオンのままにしておくことを推奨します。

8.1 状態マシンの通知

マネージャが起動すると、通知は自動的にオンになります。

状態マシンには以下のロジックが必要です。

- モートが存在しない場合は通知をオンのままにしておきます。AP 以外のデバイスに対する **moteOperational** イベントが発生するまでは、通知をオンのままにしておきます。（**moteJoin** ではなく）**moteOperational** 通知を待つと、起動するが参加を完了できないモートが引き続きネットワークを確実に検出できます。
- 「最後」のモートが参加した後に非アクティブ化します。最後の **moteOperational** イベント後に適したタイムアウト値（例：1 時間）があるはずです。
- モートが離脱したら再度アクティブ化します。**moteLost** イベントまたは **moteReset** イベントが発生したら、通知を再開します。
- タイムアウト時間内にモートの離脱またはモート参加のリセットがなかった場合は非アクティブ化します。動作時の消費電力を最小限に抑えるため、最後の **moteLost** イベントまたは **moteReset** イベント後に適したタイムアウト値（例：1 時間）があるはずです。離脱したデバイスがタイムアウト時間内に再参加しないと、通知が別の理由で再開されるまで再参加できなくなるので注意してください。
- より多くのモートをシステムに追加する場合は、通知を再開します。

9 アプリケーション・ノート：組込みマネージャの通電バックボーンを使用した待ち時間の改善

9.1 概要

すべての SmartMesh マネージャには様々な構成があり、これを使用して、モートの平均消費電力やメッセージ待ち時間などの主要なネットワーク性能基準を調整できます。ネットワークを調整して特定の問題を解決する方法は数種類あります。本書では、SmartMesh IP 組込みマネージャの通電バックボーン機能を示し、この機能をネットワーク内で使用するべき場合とそうでない場合について詳しく説明します。通常は略して単にバックボーンと呼ばれますが、このスロットフレームにより、上り方向または双方向で待ち時間の短い通信が可能になります。上り方向の通信では非通電デバイスで余計なエネルギーを消費しませんが、双方向のバックボーンを有効化すると、すべてのデバイスで電力の増加が強いられます。

制限事項:

- バックボーンを下り専用にする方法はありません。
- バックボーンのモードおよびサイズはマネージャの起動前に選択されるので、マネージャをリセットせずに変更することはできません。
- バックボーンの動作は、モート参加時に提示される電源設定によって決まります。いったん割り当てられると、参加後に電源設定を変更してもバックボーンの動作には影響しません。

本書での電力計算例は、SmartMesh Power and Performance Estimator の値を使用して行っています。

9.1.1 全般的な目的

バックボーンは ZigBee のような待ち時間の短い上り通信を実現するために開発されました。ZigBee ネットワークではルータが通電されており、すべてのデバイスが 1 つのルータの範囲内に存在する必要があります。上りバックボーンをアクティブにした SmartMesh IP ネットワーク（**bbmode** = 1 を使用して設定）では、モートが通電ノードを親に持つことを前提に、任意のモートが任意のスロットで送信可能です。

通電ノードは、それ自体がスロットで送信しない場合、子ノードからのパケットを受信します。結果として、非通電デバイスにはアイドル状態の TX イベントが大量に存在することになりますが、Eterna モートを使用する場合はエネルギーを消費しません。通電デバイスにはアイドル状態の RX イベントが大量に存在し、エネルギーを消費します。パケットが到着する場合に備えて無線を受信モードで動作状態にしておく必要があるからです。上りのバックボーンをより長いスロットフレーム長で設定できますが、ほとんどのアプリケーションの動作が最高の状態になるのは、バックボーンのスロットフレーム長が最短の 1 スロットのときであることがこの後、分ります。長さは **bbsize** パラメータによって設定され、設定できる値は 1、2、4、および 8 スロットです。

下りバックボーンも同様に機能しますが、今度は**すべての**デバイスが受信してエネルギーを消費する必要があることが異なります。双方向のバックボーンがアクティブな場合（**bbmode** = 2 を使用して設定）は、偶数スロットが上りバックボーンに使用され、奇数スロットが下りバックボーンに使用されます。このように偶数／奇数で分けると、2 方向間

でトラフィックが衝突しなくなります。これは不完全な RF 環境での呼び出し応答トラフィックでは非常に重要です。Dust コマンドのトラフィックでは下りバックボーンを使用せず、ユーザ・トラフィック用の予備にしています。サポートされている**唯一**の双方向バックボーン長は 2 スロット長です。

すべてのバックボーン動作は共有リンク上で行われます。同時に、残りのネットワークは優先度の高い専用リンク上で引き続き動作します。バックボーン・リンクはそれ以外のすべての動作とはチャンネル・オフセットが異なるので、バックボーン上のトラフィックが他のバックボーン・トラフィックと衝突する可能性がある場合でも、引き続き専用リンクで行われる信頼性の高い通信には影響しません。どのような状況であっても、バックボーンが全体の信頼性を低下させることや、ネットワークの待ち時間を増加させることはありません。

9.1.2 上りバックボーンで RX を有効化する設定

SmartMesh IP モートは、その最大定常状態電流を **powerSrcInfo** 構造体の **maxStCur** フィールドの参加要求パケットで報告します。マネージャは、モートが上りバックボーンで RX リンクを張ることができるようにするためのフラグとして最大の設定値（0xFFFF）を使用します。この最大の設定値はセンサー・アプリケーションが指定する必要があり、モートが本当に通電されているか、またはモートに大容量のバッテリ電源がある場合にのみ使用します。これは、バックボーン内に受信リンクがあるモートは 1mA を超える電流を流すことがあるからです。簡単のため、**通電**モートとして **maxStCur** = 0xFFFF が設定されているモートを調べます。すべてのモートは上りバックボーンに TX リンクを張ることができますが、RX リンクは通電モートにのみ与えられます。**setParameter<powerSrcInfo>** API の **maxStCurrent** フィールドを 0xFFFF に設定すると、ノードは「通電状態」とみなされますが、それ以外の設定では非通電となります。デフォルトでは、ノードの出荷時の設定は maxStCurrent = 0xFFFE なので、マネージャはバックボーンを呼び出さずに必要に応じてリンクを追加できます。こうしたデバイスは、最も混雑しているネットワークで平均電流が 500μA 未満になると思われます。通電ノードの際立った動作は、上りバックボーン・スロットフレームで受信リンクを取得することであり、それ以外には通電ノードと非通電ノードのスケジュール間に差はありません。

9.2 アプリケーション 1：低待ち時間アラーム

ネットワーク

- バックボーン・モード **bbmode**：1
- バックボーン・スロットフレーム・サイズ **bbsize**：1

予想平均性能基準

- 待ち時間：約 15 ミリ秒/ホップ
- リーフ・ノードでの割増電力：なし
- ルーティング・デバイスでの割増電力：925μA（1 スロットのバックボーン）
- 全トラフィック限度：約 45 パケット/秒

低待ち時間アラームのネットワークでは、各非通電モートが通電モートの範囲内にあることを確認する必要があります。その後、通電モートは**通電**バックボーンを形成し、どのモートもこのバックボーンから最大で 1 ホップの距離になります。そのため、すべてのモートは任意のスロットで送信できます。送信が失敗した場合、バックボーン・スロットが共有されているので、送信側のモートは失敗の原因がパケットの衝突であったとみなします。モートには、この競合

ベースの一連のリンクを正しく使用するために、指数関数的に増加するランダムなバックオフがあります。バックボーンが AP から複数ホップの位置に広がるのを妨げるものはありません。平均的な安定性とバックオフの仕組みを考慮すると、ホップ当たりの平均待ち時間は約 15 ミリ秒、つまり 2 スロットになります。ただし、バックボーンは衝突する可能性があるので、最も厳しい場合に並列の専用リンクによって実現される待ち時間をアプリケーションが許容できる必要があります。

ルーティング目的の場合は、専用の上りリンクとバックボーン・リンクが均等に扱われます。あるモートに上りパケットがあり、次のスロット内にあるこのモートの親への専用上りリンクもこのモートに存在する場合、このモートはパケットを共有バックボーン・リンクではなく専用リンクで送信し、親も専用リンクで受信します。こうなると必然的にチャンネル・オフセットが異なるので、バックボーン・リンク上で送信しようとする同じ親の別の子が最初の送信と衝突することはありません。同様に、モートに別のリンク（参加通知受信リンクや下り RX リンク）がある場合、モートによる処理は、上りバックボーン・リンクに対してではなく、この別リンクに対して実行されます。また、複数のホップを進むパケットは最終的に専用リンク上のいくつかのホップとバックボーン・リンク上のいくつかのホップを進むことになります。こうしたことはすべて、モートにとってどちらのリンクが先に使用できるかと、バックオフが発生するかどうかのランダムな結果として行われます。

モートのバックボーンの親は一度に 1 つだけですが、これは上り方向の親（通常は 2 つ）のうちの 1 つになります。バックボーン・リンクは経路障害を調べる目的には使用されませんが、経路障害がバックボーンの親への専用リンク上で検出された場合、モートは 2 番目の親へのバックボーンの使用を自動的に開始します。専用リンクはこの 2 番目の親までずっと使用されていましたが、この場合もアプリケーションは経路障害中に長い待ち時間に耐えられる必要があります。

上りバックボーンは、あまり多くのデータは送信しないがパケットの生成時は待ち時間を短くする必要があるネットワークにとって最も有効です。平均待ち時間の要件が 100 ミリ秒未満で 30 モートのネットワークでは、専用リンクだけでこれを実現するのに十分なスロットが AP に存在しません。電力が心配な場合は、長めの **bbsize** 設定である 2、4、または 8 スロットを使用できます。これにより、それ相応にルーティング・デバイスの消費電力が減少し、待ち時間が長くなります。

9.3 アプリケーション 2：呼び出しと応答

ネットワーク

- バックボーン・モード **bbmode**：2
- バックボーン・スロットフレーム・サイズ **bbsize**：2

予想平均性能基準

- 往復の待ち時間：約 60 ミリ秒/ホップ
- リーフ・ノードによる割増電力：462μA
- ルーティング・デバイスでの割増電力：925μA
- 全トラフィック限度：約 16 パケット/秒（大半のモートが 1 ホップである場合）

パケットを単一の宛先に送信する場合、デフォルトの SmartMesh IP ネットワークはパケットの進入（下り）をスロットフレームにつき 1 つに制限します。安定性が不十分な場合、これは 2 秒当たり 1 パケットに満たない値です。高速な下りが要求されるアプリケーションでは、双方向のバックボーンを使用できます。上りバックボーンとは反対に、下りバックボーンはルーティング相当ではありません。内部コマンドのトラフィックは下りバックボーン・リンクを使用せず、下りバックボーンがアクティブのとき、ユーザ・トラフィックは下りバックボーンだけを使用するよう制限されます。

この双方向バックボーンはすべてのデバイスで 462µA 以上を消費し、デバイスがたまたま上りルータであった場合は 462µA を追加で消費します。経験則から、パケットは AP からモートまで進むことが可能なはずであり、応答は約 60 ミリ秒/ホップ以内に戻るはずです。バックボーンは共有帯域幅を使用するので、応答を待ってから次の呼び出しを下流に送信することを推奨します。60 ミリ秒/ホップという値は平均時間であり、アプリケーションは最大 2 秒までの外れ値を許容できる必要があります。

9.4 アプリケーション 3：すべての低トラフィック・ネットワークでの短い待ち時間

ネットワーク

- バックボーン・モード **bbmode**：1
- バックボーン・スロットフレーム・サイズ **bbsize**：1

予想平均性能基準

- 1 ホップのモートからの待ち時間：約 15 ミリ秒
- モートでの割増電力：なし

AP が電力制限デバイスでない場合は、上りバックボーンを使用して、モートでの消費電力を増加させずにネットワーク全体の待ち時間を短縮できます。このためには、上りバックボーンを起動して、AP に唯一の通電デバイスとしてマークを付けます。その結果、AP の 1 ホップの子モートだけがバックボーン TX リンクを張ります。これらのリンクでは、モートを使用しない場合、エネルギーが消費されないことに留意してください。パケットを生成するか子モートからパケットを受信する任意の 1 ホップ・モートは、後続のスロットの AP にパケットを転送できます。マルチホップ・ネットワークでの待ち時間は、ネットワークの最初のホップまでは同じ時間のままですが、すべてのパケットが 1 ホップのリングを通過する必要があるので、上りバックボーンがアクティブになった場合は、すべてのモートで平均待ち時間が減少します。

ネットワークに大量のトラフィック（>合計 15 パケット/秒）がある場合は、バックボーン・リンクに競合が発生し、伝送は衝突します。これらの衝突に起因する再試行数の増加は、1 ホップ・モートでの消費電力に影響しますが、これによってスループットが減少することはなく、待ち時間は増加しません。電力の影響を受けやすいアプリケーションの場合は、この 1 ホップ・バックボーンの使用を、トラフィックが少ないと分っている場合に限定することを推奨します。

9.5 バックボーンの不適切な使用 1:専用リンクの置き換え

ネットワーク

- バックボーン・モード **bbmode**:1
- バックボーン・スロットフレーム・サイズ **bbsize**:8

予想平均性能基準

- リーフ・ノードでの割増電力:なし
- ルーティング・デバイスでの割増電力:116µA

上記アプリケーションをすべて読むと、あらゆるものにバックボーンを使いたくなる可能性があります。ただし、バックボーンが一連の等価な専用リンクより消費電力が多く待ち時間が長い場合がいくつかあります。例えば、ネットワークには本当に通電されているモートが存在しないが、上りの待ち時間を、専用リンクだけを使用するネットワークでの観測結果より少し短縮したい場合を考えます。そこで、子モートを持つすべてのモートで消費される余分な 116µA を許容できると考えて、すべてのモートが実際に「通電状態」であると報告し、上りバックボーンのサイズを 8 スロットに増やすことにします。

この場合の上りバックボーンについて効率が悪い部分は、子モートを持つモートが、バックボーン・リンクまで AP と同じ回数(厳密には 8 スロットに 1 回)を受信する必要があることです。カスケード接続された専用リンクを割り当てると、リンク数が AP と同数になるモートはなくなります。これはアイドル RX リンクではなくアイドル TX リンクなので、高効率のネットワークはできるだけ多くの負担を AP に負わせます。消費電流が 116µA の場合は、より高速な基本帯域幅で同じネットワークを構築することが可能で、全体の待ち時間が短くなります。また、更にいいことには、アプリケーション 3 で説明したやり方でこのネットワークでの待ち時間を追加コストなしでより短縮できます。

実際に、長さが最短ではない上りバックボーンを使用しようとする場合は、オプションを慎重に検討してください。

9.6 バックボーンの不適切な使用 2:トラフィックの多いネットワーク

ネットワーク

- バックボーン・モード **bbmode**:1
- バックボーン・スロットフレーム・サイズ **bbsize**:1
- アクセス・ポイントまでの全トラフィック:30 パケット/秒
- 100 モート

予想平均性能基準

- ノードでの割増電力:最大 120µA

専用リンクはバックボーン・リンクより優先順位が高いことを思い出してください。ネットワークが混雑している場合、アクセス・ポイントはほとんどのスロットに専用リンクを張って上りトラフィックを受信します。モートはアクセス・ポイントの全スケジュールを知らないので、アクセス・ポイントが専用リンク上での受信で混雑しているバックボーン・リンク上でモートは送信して失敗をすることが多くなります。このようにバックボーンの帯域幅で競合しているいくつかのモートがある場合は、バックオフがある場合でも、ほとんどのバックボーン送信が失敗します。モートがフルサイズのパケットを送信している場合は、これらのすべての障害によって、余計な送信試行で最大 120µA を消費することになります。これはルーティング・デバイスにとっては大きな問題にはなりませんが、全消費電流が 30µA と予想される低消費電力デバイスにとっては重大になることがあります。このシナリオはネットワークが大きくなるにつれて現れる可能性があります。報告頻度の低い少数のモートから始めるネットワークでは、バックボーンによって待ち時間が大幅に改善されますが、高速報告モートが追加されるので、これら元のモートに対する待ち時間のメリットは減少し、元は空きだったバックボーン・リンクは繰り返し失敗するので、多大な犠牲を強いられるようになる可能性があります。こうしたシナリオでバックボーンを非アクティブ状態にするために、ネットワークをリセットする必要があることにも注意してください。

アプリケーション 3 で説明したように、15 パケット/秒という閾値は、バックボーンが有益であるか問題となるかを判断する良い目安です。この場合も、バックボーンをアクティブにすることでパケット待ち時間が増加したり、この混雑したネットワークで信頼性が低下したりすることはなく、影響を受けるモートの消費電力が増加するだけです。

バックボーンの障害はパケット障害として数えられるので、過剰に使用されたバックボーン・ネットワーク内のモートによって報告された経路安定性の値は非常に低くなります。安定性の値が 50%より低くなるかどうか注意して、この問題を診断するのに役立ててください。これらの対象経路は、「RSSI-安定性」滝型曲線を引くと最も明確になります。

10 アプリケーション・ノート：深い IP ネットワークの構築

10.1 概要

デフォルト設定では、ソース・ルーティングされたパケットのタイムアウトが原因で、SmartMesh IP ネットワークの範囲は約 8 ホップになります。ただし、いくつかの変更（および十分な接続性）により、SmartMesh IP ファミリは最大 32 ホップの長い直線距離に及ぶネットワーク（見通し線配置で約 10Km）を十分に構築できます。これには、センサーが出口点から同じ方向に離れて配置される傾向がある、坑道内での環境のモニタや伝送線モニタなどのアプリケーションが含まれます。1 次元の配置を考慮すると、マネージャから離れた場所にあるワイヤレス・センサー・ノード（「モート」）からのパケットは、宛先に到達するのにより多くのホップが必要になります。これらのモートは、マルチホップ・ネットワーク内で「深い」状態にあると呼びます。密度の高いメッシュ・ネットワークと比較して、こうした深いマルチホップ・ネットワークに特有の性能特性がいくつか存在します。

1. ネットワークに完全に参加するのに時間がかかります。
2. マネージャは全トラフィックの下限値をサポートできます。10.5 パケット/秒という全放出の最大値を考慮する必要があります。
3. 配置の場所と接続性にはより注意が必要です。
4. パケットの待ち時間はネットワークの深さに応じて長くなります。

10.2 配置のガイドライン

SmartMesh のすべての配置と同様に、モートは他の 3 つの潜在的な親の範囲内にすべて配置して、ネットワークの信頼性を確保する必要があります。線形配置の場合、これはすべてのセンサー・モートについて、範囲内でマネージャに近い方に 3 つのモートが存在する必要があるという意味です。更に、この性質をマネージャの近くで保存するため、マネージャの近くにいくつかの中継器モート（センサーを内蔵するモートまたは内蔵しないモート）を配置することを推奨します。目的の環境での無線範囲を **R** とすると、配置は図 1 に従って実行する必要があります。

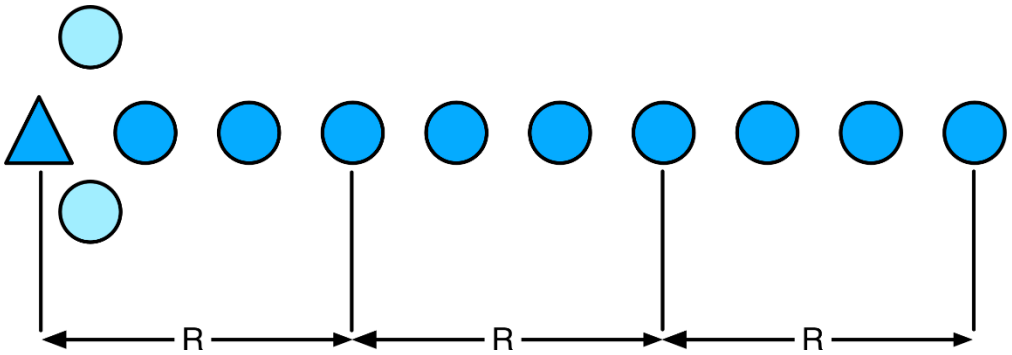


図 1. 各センサー・モート（濃青色の円）には、マネージャ（三角）の近くに 3 つの上り隣接モートがあります。中継器（淡青色の円）は、トラフィック処理と空間的多様性を実現します。

実際の配置での検出点がこれより少ない場合は、中継器を追加して、デバイスごとに 3 つの潜在的な親という要求密度を確保する必要があります。有効範囲内にすることができる総距離は、範囲に影響する配置環境に大きく影響

されます。図 1 に示すように、例えば **R**=100m である場合、100 モートのネットワークは外側に 3km を超える距離までモニタできます。デバイスを地表から 20～30 メートルの高さで見通し線内に配置すると、この範囲を 5 倍（つまり 15km 超）に延ばすことが可能であり、デバイスを地表から 1 メートルの坑道に配置すると、範囲は半分（1.5km）に縮まる可能性があります。

10.3 範囲の割り出し

坑道と伝送線という 2 つの配置例は、これらの非常に異なる環境での無線伝播特性のため、対照的なデバイス範囲に位置すると予想されます。これらの設定のいずれの場合も、ワイヤレス・センサーの実際の完成品または試作品を使用し、使用する予定のアンテナを付けた完全一体型の 1 対のモートによってデバイス間の範囲を直接測定する代替手段はありません。この測定範囲は、中継器の数とネットワークの最大距離の両方を通知します。この理由から、OEM 供給先は開発サイクルのできるだけ早い段階に範囲測定を実施することを推奨します。

範囲の推定の詳細については、アプリケーション・ノート配置の計画を参照してください。

10.4 モートとマネージャのバージョンおよび設定

深いネットワークを構築するには、SmartMesh IP モートのバージョン 1.3 以降が必要です。アップグレード処理が必要な場合は、配置の前に実行しておく必要があります。また、マネージャは従来の配置と異なる構成変更をいくつか行う必要があります。これらの設定もネットワークの構築前に行って、持続させる必要があります。

VManager の CLI 設定：

```
su becareful
set config network numParents=3
set ini RLBL_BCAST_TO=240000
set ini RLBL_FLOOD_TO=240000
set ini RLBL_MAX_TO=240000
set ini MINLINKS=8 set ini NUMMCAST=0
set ini RLBL_JOIN_TO_M=27
set config network topologyType=EVENT
set config network usFrameSize=256
set config network dsFrameSize=512
```

組込みマネージャの CLI 設定（バージョン 1.2.1 以降が必要）：

```
su becareful
set config numparents 3
seti ini rlblbcto_f 240
seti ini rlblskto_f 240
seti ini rlblmaxto_f 240
seti ini rlblskto_s 240
seti ini minlinks 8 (refer to the calculation of L in the "Calculating Links" section)
seti ini iscascading 0
seti ini nummlinks 0
```

モートでは構成変更の必要はありません。

10.5 リンクの計算

割り当てられるリンクの数は（上記で詳述した設定の場合と同様に）、最小でも 8 に設定する必要があります。ネットワークでの報告率を高くしたり待ち時間を短くしたりするには、リンク数を増やすことが必要な場合があります。リンク数は次式を使用して計算できます。

L = [1.8M/T]

ここで、リンクの数は **L**、ネットワーク内でのモートの数（中継器を含む）は **M**、レポート間隔は **T** です。

角かっこは、ここでは端数の切り上げが必要であることを示しています。全ネットワーク放出を 10.5 パケット/秒に制限して、100 モートで最大限のスループットが得られるようにすることを推奨します。これは 1 モート当たり 10 秒につき 1 パケットであり、健全性レポートのパケットを伝送するには若干の追加となります。この場合のリンク要件を計算すると、**L**=18 となります。

メモリに制限があるので、積 **LM** の最大値は 1800 に制限します。例えば、100 モートの深いネットワークの場合は、**L**=18 という値が許容最大値です。50 モートの小規模ネットワークでは、パケットの待ち時間を最小限にする場合、**L**=36 と設定できます。すべての SmartMesh ネットワークと同様に、リンク数の追加と平均エネルギー消費量とのトレードオフに留意することが必要です。

10.6 待ち時間の推定

ネットワーク内の各モートの待ち時間分布は、正確に予測するのが困難です。組込みマネージャで 100 個のモートをテストして得られた図 2 に示すモート別待ち時間の例について考えます。待ち時間は一般的にモートの深さに応じて長くなりますが、待ち時間の測定値には大きなばらつきがあり、特に最大の測定値で顕著です。グラフから得られた一例として、最も深いモートの場合、待ち時間の中央値は 10.2 秒であり、待ち時間の 99 パーセンタイル値は 20.6 秒です。

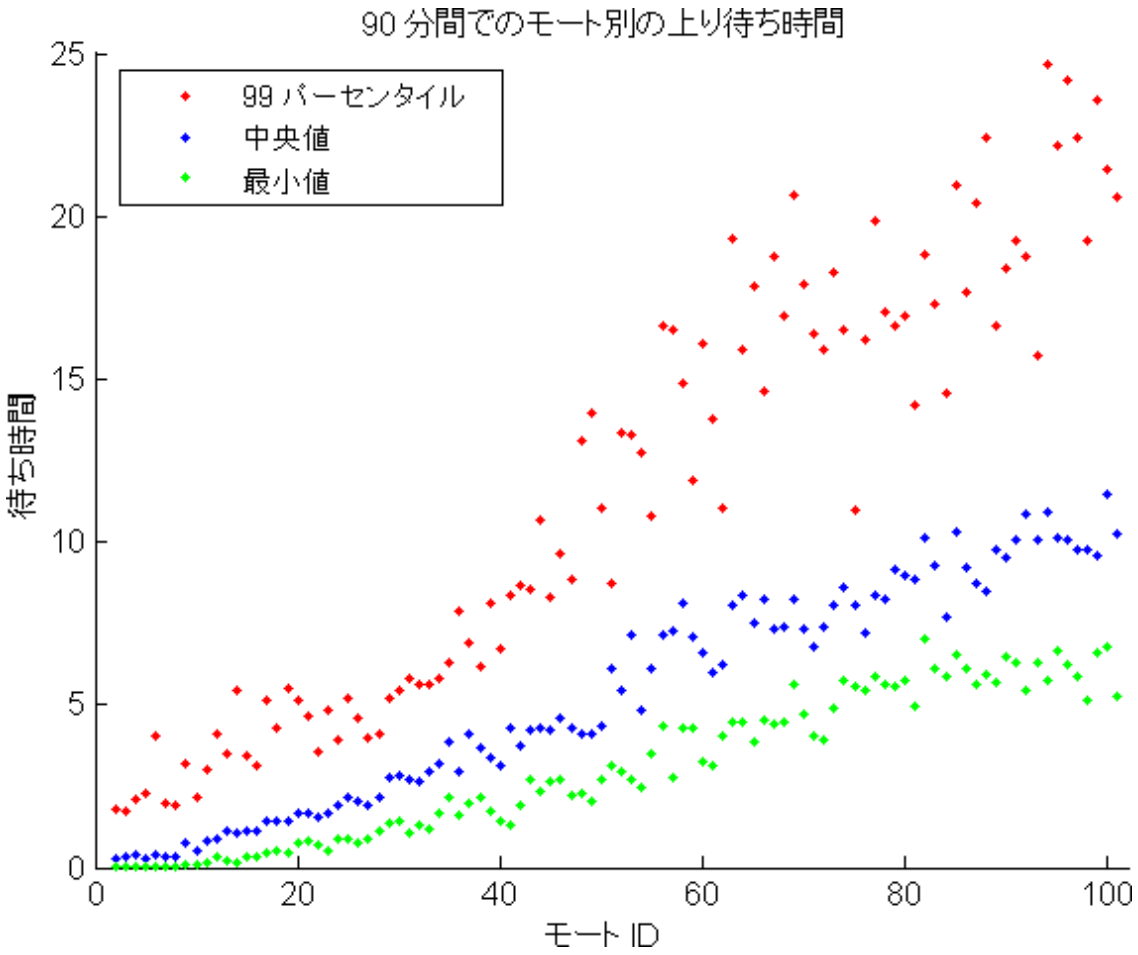


図 2. モート ID の関数としての待ち時間。デフォルトの設定で 1 時間にわたってデータを収集した場合。最大値を赤の点、中央値を青の点、最小値を緑の点で示します。このネットワークでは、平均経路安定性の測定値は 85%でした。

このデータを使用して、ネットワーク内で最も深いモートの待ち時間を推定できます。この状況では、平均経路安定性 **S** が重要なパラメータです。**S** が示していることは、様々な RF 関連の要因が原因で、すべての送信が最初の試行で宛先に到達するとは限らないという考え方です。例えば、100%の経路安定性とは、すべてのパケットが最初の試行で宛先に到達するという意味であり、80%の経路安定性とは、パケットの 8/10 は最初の試行で宛先に到達し、2/10 は自動的に再送信されるという意味です。

<median(latency)> = 0.75M/LS

図 2 の赤い点は、待ち時間の予想できる 99 パーセンタイル概算値です。したがって、待ち時間の 99 パーセンタイル値は、中央値の 2 倍から 3 倍の間になると予想できます。経路安定性が低い場合は、パケットがモートのキューで作成されるので、待ち時間分布の裾がより長くなる可能性があります。満杯になったキューの影響の特定および軽減について詳しくは、アプリケーション・ノート:「混雑したネットワークのデバッグ」を参照してください。

10.7 距離のカバー

複数の組込みマネージャ・ネットワークを使用して長い直線距離をカバーする必要がある使用事例には、いくつかの種類があります。例えば、200m ごとにモートが配置された 100km の伝送線について考えてみましょう。この伝送線は 5 つの 100 モート・ネットワークで監視することができます。この設定では 2 つの選択肢があります。

1. マネージャに適した場所が簡単に見つかる場合、50 モートで構成される 2 つの直線グループの間にマネージャを 1 つ配置します。この方法では、100 個のモートを 1 列にする場合と比べて待ち時間を 1/2 に短縮できるか、またはネットワークの消費電力を削減できるというメリットがあります。この場合、マネージャとマネージャの間には 100 モート分の距離があります。
2. 電力やインターネット接続の観点からマネージャに適した場所を見つけるのが難しい場合、2 つのマネージャを同じ場所に配置して 100 モートのネットワークを互いに反対の方向に接続します。この場合、マネージャ・ペアとマネージャ・ペアの間には 200 モート分の距離があります。

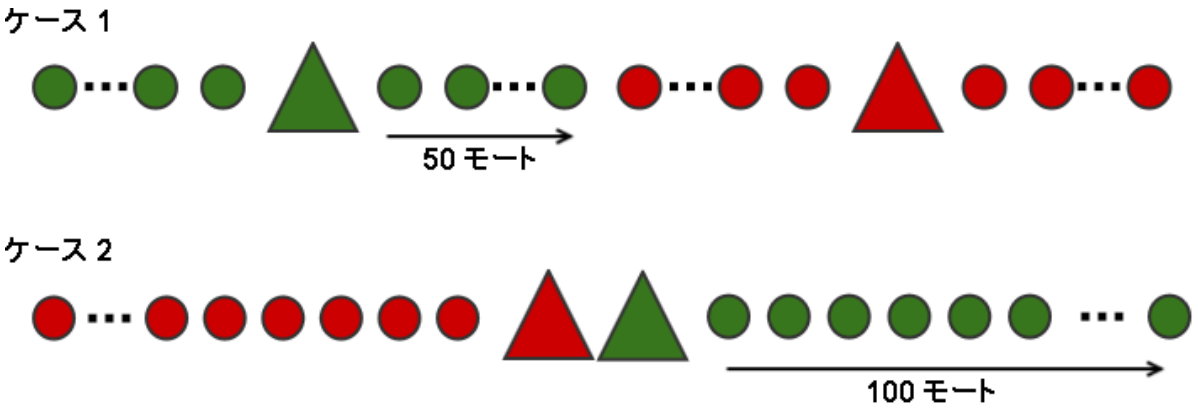


図 3. ネットワークを色分けした 2 つの配置戦略。(上)待ち時間または消費電力を最小化する配置、(下)マネージャ・サイトの数を最小化する配置。

ケース 1 の性能を見積もる場合、マネージャから最も遠いモートが 50 モートなので、待ち時間とリンクの計算で **M** = 50 を設定します。ただし、積については、2**LM** < 1800 となるようにします。ケース 2 の性能を見積もる場合、すべての計算で **M** = 100 を設定します。

11 アプリケーション・ノート: 重複ネットワーク

11.1 概要

特定の配置では、複数の SmartMesh IP ネットワークにまたがる数百または数千のモートが小さな領域内で必要になることがあります。SmartMesh プロトコルは信頼性の要件として時間同期を重視するので、重複する空間に置かれている複数のネットワークでこれらのモートを配置するのは危険を伴うと考えられます。このアプリケーション・ノートで示すように、基本経路安定性は十分高いと仮定すると、重複ネットワークを配置する大きな危険はありません。最大の影響、つまりすべての重複ネットワークにわたる有効な経路安定性の低下を数値化します。1 つの**無線空間**を、1 台のワイヤレス・デバイスによる伝送でカバーされる領域として定義します。複数のデバイスが同じ無線空間にある場合、各デバイスがこの空間内にある任意のデバイスから伝送を受信できると同時に、これらのデバイスからの干渉の影響を受けることになります。同じ無線空間にある 2 つの異なるデバイスから、同時に同じ伝送周波数で 2 つの伝送が行われた場合、これらの伝送は**衝突**します。

実世界の重複ネットワーク例として、Dust Networks では、10～30 のネットワークに含まれる 1000 モートが同じ無線空間内で同時に稼働する環境で、すべてのテストを実施しています。このようなネットワークでは伝送が衝突しますが、全体的なデータ信頼性の質が低下することはありません。

11.2 方法

複数の組込みマネージャの SmartMesh IP ネットワークに、それぞれ 100 モートの全容量が与えられる配置について検討します。こうした 10 のネットワークがすべて同じ無線空間内にあり、そのうち 1 つのネットワークから 1 つのモートを選ぶとします。SmartMesh マネージャの帯域幅割り当てアルゴリズムを使用しているので、このモートからの伝送が、同じネットワーク内にある他の 99 モートからの伝送と衝突することはありません。ただし、重複ネットワークのいずれか 1 つからの伝送とは衝突する可能性があります。復号が関係しているので、どちらかより早い段階で開始した伝送が正しく受信する可能性が高く、どちらか途中で開始した伝送は再送信を要求される可能性があります。

各ネットワークでのモートのレポート頻度に基づいて、そのネットワークの単位時間当たりの伝送の総数を計算できます。また、この伝送の総数に基づいて、選び出した 1 つのモートと衝突する確率を計算できます。衝突によってモートがその伝送に失敗した場合、モートは次の割り当てリンクで伝送を自動的に再試行します。再試行のたびにモートの経路安定性測定値は低下しますが、データは次の割り当てリンクでとにかく伝送されるので、データの信頼性は維持されます。

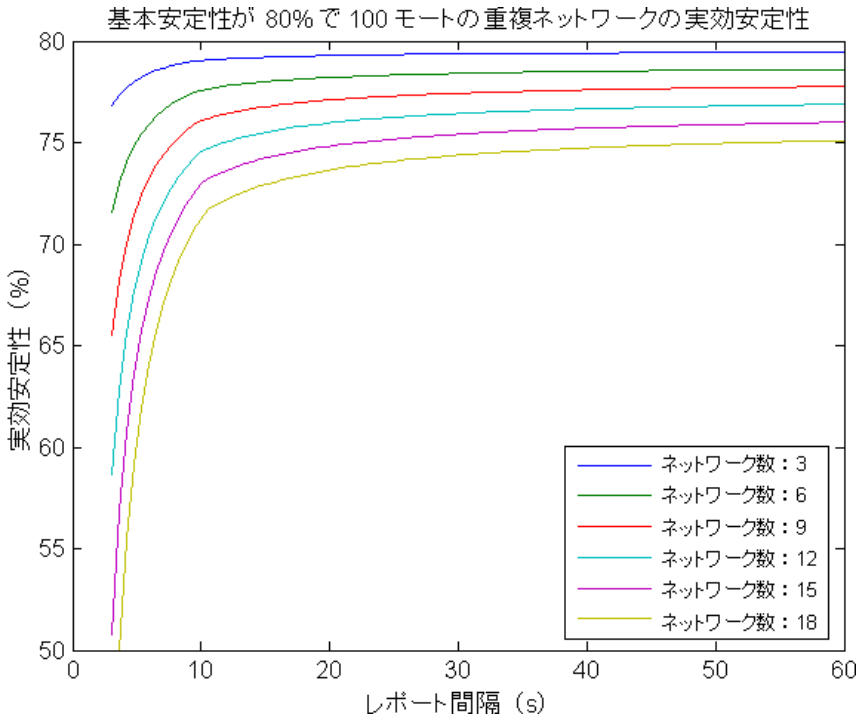
100 モートの単一ネットワークを配置して経路安定性を測定した場合、結果は 80%になることがあります。これがいわゆる**基本経路安定性**です。重複ネットワークの数が 10 になると、安定性は低下していきます。重複ネットワーク内のトラフィックが増えると、安定性はより低下します。この最終値を**実効経路安定性**と呼びます。次のセクションでは、様々な数の重複ネットワーク、様々な報告率、様々な基本安定性を想定して、実効安定性を計算します。SmartMesh ネットワークは、正しい受信には伝送ごとに平均 1 回の再試行が必要になる場合でも、その動作が完全なデータ信頼性を実現するように設計されています。

一般に、実効経路安定性が 50%未満のとき、ネットワークを稼働することは推奨しません。このような安定性状況で動作するネットワークは、マネージャのトランスポート層の障害により、モートのリセットが増える傾向にあり、データ信頼性が低下する可能性があります。50%未満の信頼性で動作するネットワークを構築する場合、SmartMesh マネージャで割り当てる帯域幅を大幅に抑制することで、ネットワーク・スループットを制限してモート性能を高める必要があります。

11.3 結果

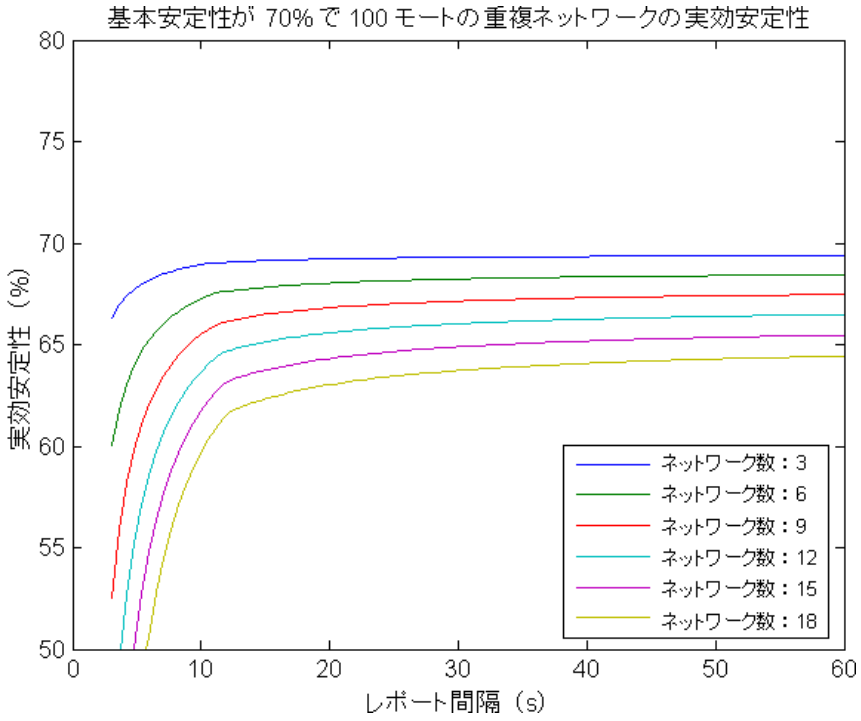
この計算では、配置の最も厳しい場合を想定しています。第 1 に、すべてのモートおよびマネージャは同じ無線空間にあるものとします。第 2 に、すべての伝送は、できる限り長く続く 90 バイトの最大アプリケーション層ペイロードであるとします。第 3 に、各ネットワークには最大の 100 モートがあります。また、重複ネットワーク内にあるすべてのモートが同じ速度でデータを報告しているとします。低速帯では、モートは 60 秒間隔でデータを報告します。高速帯（組込みマネージャ SmartMesh IP ネットワークのパケット/秒制限値近く）では、モートは 3 秒間隔でデータを報告します。

80%の基本安定性から始めて、安定性が損なわれ始めるまで同じ空間に多くのネットワークを許容できることが分ります。



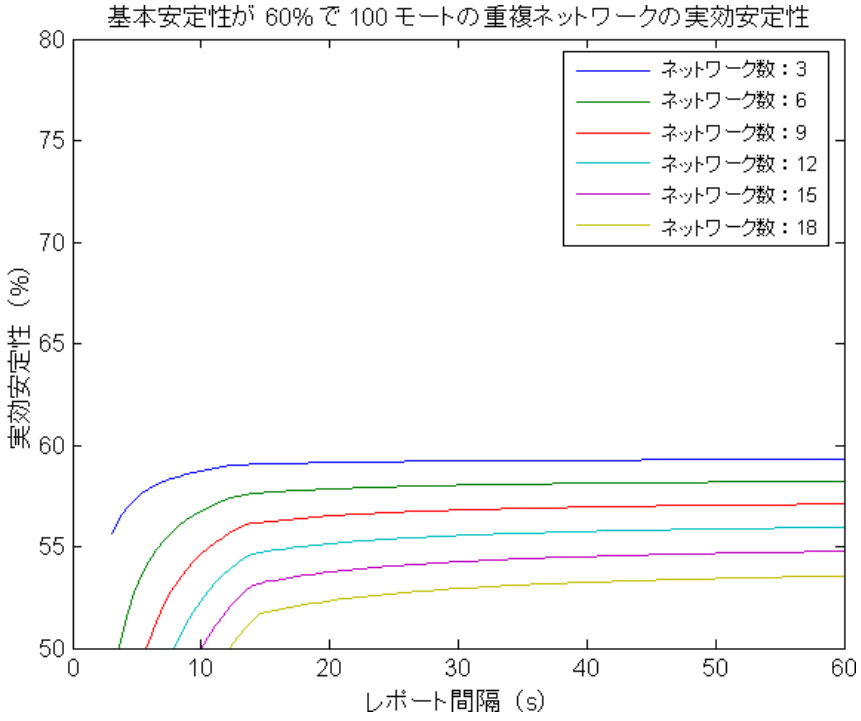
このグラフから、80%の基本安定性を確保している場合は、15 の重複ネットワークをその全容量で稼働（つまり、各モートが 3 秒おきに 1 パケットを報告）しても、実効安定性が 50%を下回ることはありません。実効安定性の低下に見合った量の消費電力と待ち時間の増加が生じますが、信頼性は確保されます。

基本安定性が 70%の場合に同じ分析を繰り返すと、12 以上の重複ネットワークで何らかの報告限界が示されます。



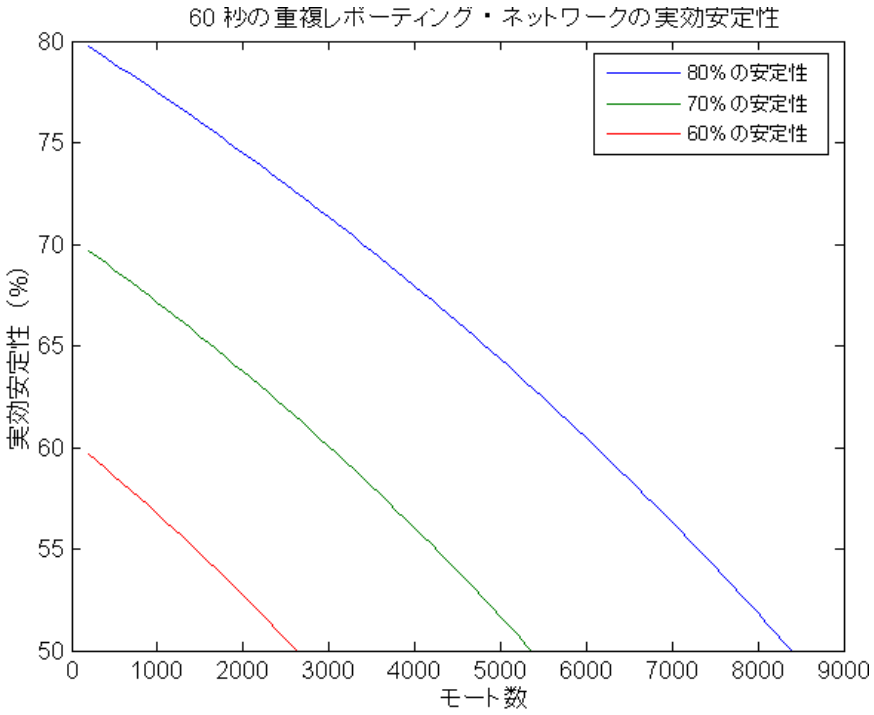
これらの限界に達するには、種々の曲線が 50%の実効安定性の軸と交差する場所を調べます。各モートが 3 秒ごとに 1 パケットを報告する場合、9 つのネットワークを同じ無線空間で稼働することは推奨ガイドラインの範囲内です。

次に、同じ軸にプロットされた 60%の基本安定性を考察すると、推奨ガイドライン内での信頼できる動作に対するマージンが大幅に少なくなっています。



この場合、3 つのネットワーク以外のすべてのネットワークの実効安定性が 50%を切る可能性があります。

こうした結果を得ることができる方法がもう 1 つあります。報告回数が最小限で 60 秒間隔のモートがある場合、同じ無線空間に安全に配置できるモートの数はいくつでしょうか。



基本安定性が 60%、70%、および 80%の曲線群を上の図に示します。高い基本安定性から始めた場合は、1 つの無線空間で数千のモートをサポートすることが可能です。こうした高モート密度では、存在するネットワーク数は実際のところ問題ではありません。例えば、5,000 モートを配置できる場合は、50 モートのネットワーク数が 100 であっても、100 モートのネットワーク数が 50 であっても変わりません。存在するトラフィックの量、したがって干渉の量は同じです。

11.4 結論

前のセクションのグラフから、いくつかの結論を出すことができます。

- 最大 1,800 のモートが 18 のネットワークにわたって広がっている、平均発行間隔が 15 秒につき 1 パケットの実装形態は、どれも推奨ガイドラインに適合します。実効経路安定性が 5%～10%低下することが予想され、その結果として消費電力と待ち時間が若干増えます。どのくらい増えるかを数値化するには、SmartMesh Power and Performance Estimator を使用します。
- 100 モートの単一ネットワークを試験的に配置して基本安定性を測定し、ネットワークの信頼性を維持する上で安全な重複の程度およびトラフィック・レベルを調べることを推奨します。

干渉の影響をより軽減するには、以下の手順を実行できます。

- ペイロードが満杯にならない状態のパケットを送信している場合は、センサー・プロセッサで満杯のペイロードが待つようになるまでデータを蓄積してからデータをモートに送信します。こうすると待ち時間は長くなりますが、衝突を発生させる可能性がある無線オン時間は全体的に短くなります。より長い待ち時間をアプリケーションで許容できる場合、これはモートの消費電力を抑えるのに優れたポリシーです。
- モート当たりの親の数を 2 から 3 に増やします。こうするとモートの経路多様性が広がり、いくつかの経路に障害が発生した場合、モートがリセットされる可能性が低くなります。ただし、親モートで消費電力の増加が必要になります。SmartMesh マネージャ CLI から以下のコマンドを実行します。

```
> set config numparents=3
> reset system
```

- ネットワークが低トラフィックのネットワークである場合は、モート当たりの最小リンク数を増やします。こうすると、モートの伝送スケジュールがランダム化され、永続的な衝突が防止される一方、上記と同様に親の増加によって消費電力が増えます。SmartMesh マネージャ CLI から以下のコマンドを実行します。

```
> su becareful
> seti ini minlinks=4
> reset system
```

- 異なるネットワークのアクセス・ポイント・デバイスをまとめて配置する場合は、常時 1 メートル以上離す必要があります。

伝送電力を低くして重複ネットワークの性能を向上させることは推奨しません。こうすると、必要な信号を含めてすべての信号が同じ大きさだけ小さくなるからです。更に、環境に周囲 RF ノイズがあると、基本安定性も低下することになります。

12 アプリケーション・ノート：モート・パラメータのリモート読出し方法

このアプリケーション・ノートでは、一部のコマンドに無線通信アクセスできる IP モート機能について説明します。**netGetParameter**（id=0x02）コマンドを使用すると、各種モート・パラメータを問い合わせることができます。パラメータの識別子と形式は、SmartMesh IP Mote Serial API Guide に記載されているものと同じです。

> ここに記載する要求パケットと応答パケットはどちらも、ベストエフォート伝送を使用して送信されます。このため、ネットワーク状況や RF 状況によっては、パケットが消失する可能性があります。アプリケーション側で、適当なタイムアウトを過ぎても応答を受信しない場合、要求を再送信するといった対応を取る必要があります。

12.1 要求パケット

モートにリモート・コマンドを送信する場合、アプリケーションで組込みマネージャの **sendData**（0x2C）API コマンドを使用するか、VManager の **POST /motes/m/{mac}/dataPacket** API コマンドを使用して、以下の値を指定します。

パラメータ	タイプ	説明
macAddress	MAC_ADDR	宛先モートの MAC アドレス。
priority	INT8U	パケットの優先順位。低（0x00）の使用を推奨します。
srcPort	INT16U	0xF0B8～0xF0BF の任意のユーザ・ポート
dstPort	INT16U	0xF0B1（2 番目のマネージャ・ポート）
options	INT8U	0x00
data	INT8U[]	送信しようとする特定の要求に含まれるペイロード・データ（以下に説明）

sendData/dataPacket マネージャ API の「data」フィールドの形式は、以下のとおりです。

コマンド ID	要求の長さ	要求データ
1 バイト	1 バイト	0～n バイト

例えば、**joinDutyCycle（0x06）**パラメータを要求する **netGetParameterCmd** コマンドを送信する場合、ペイロードは以下のようになります。

コマンド ID	要求の長さ	要求データ
0x02（コマンド=netGetParameterCmd）	0x01	**paramId** 0x06

12.2 応答パケット

応答を受信する場合、組込みマネージャに接続されたアプリケーションは、最初にデータ通知（タイプ 0x4）を **subscribe** するか、または VManager の **GET /notifications** API で **data** フィルタを指定する必要があります。応答は通知として受信され、以下の値を含むと想定されます。

フィールド	タイプ	一覧	説明
notifType	INT8U	通知タイプ	0x04（データ）
timestamp	UTC_TIME_L	なし	モートでパケットが生成された時刻
macAddress	MAC_ADDR	なし	生成モートの MAC アドレス
srcPort	INT16U	なし	0xF0B1
dstPort	INT16U	なし	対応する要求で使用された送信元ポート
data	INT8U[]	なし	応答のペイロード

応答として返される通知の「data」フィールドは、以下のような形式になります。応答データは常に RC バイトで始まり、その後にオプションの応答情報が続きます。

コマンド ID	応答の長さ	応答データ
1 バイト	1 バイト	RC（1 バイト） || 0～n バイトの追加応答データ

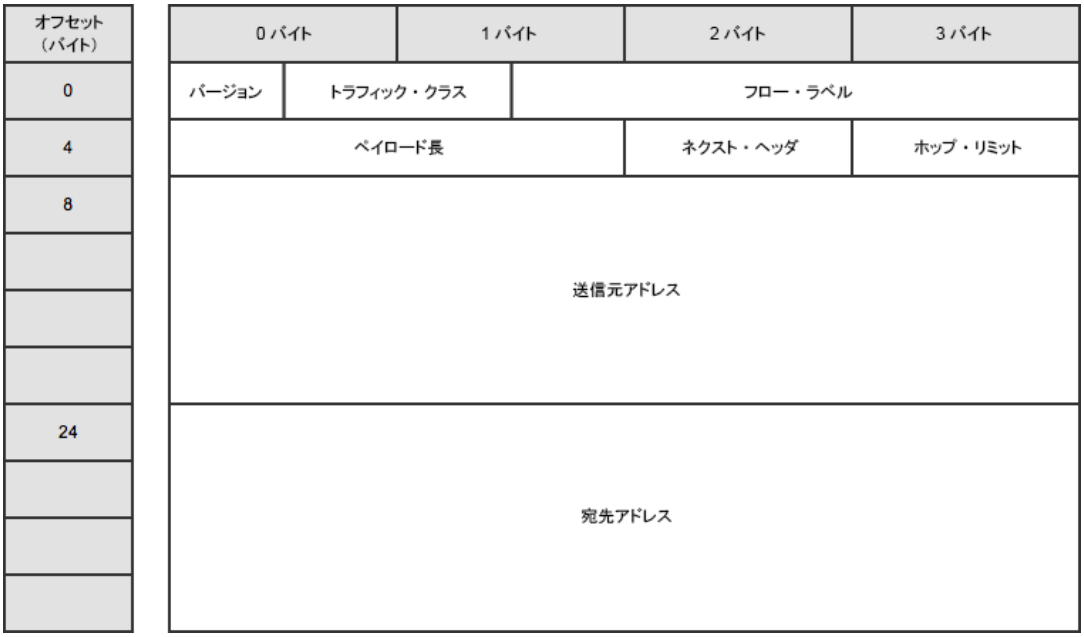
例えば、**joinDutyCycle** を取得する応答データは次のようになります。ここで、モートから返された **joinDutyCycle** は 0x7F です。

コマンド ID	応答の長さ	応答データ
0x02（コマンド =netGetParameterCmd）	3	**RC** / **paramId** / **joinDutyCycle** 0x00 / 0x06 / 0x7F

13 アプリケーション・ノート：SmartMesh IP ネットワークでの 6LoWPAN とルーティング

SmartMesh IP 製品は、RFC4944 で定義され、RFC6282 で更新された 6LoWPAN ヘッダ圧縮スキーマを使用します。これにより、IPv6 ネットワークおよびトランスポート層のヘッダを効率的に伝送できますが、続きがあります。IPv6 ネット・ヘッダは、40 バイトの固定長部分に続いて、可変長の拡張ヘッダが多数続く場合があります。固定ヘッダには、サービス品質およびネットワーク管理（トラフィック・クラス、フロー・ラベル）、エンド to エンドのアドレス指定（送信元および宛先の IPv6 アドレス）、次の拡張ヘッダの説明（ネクスト・ヘッダ）が含まれます。6LoWPAN では、各フィールドを 1～3 ビットに符号化し、完全な省略（追加バイトなし）からフィールド・インライン全体の伝送（アドレスを表す 16 バイトなど）までの圧縮レベルを指定することで、このヘッダをわずか 2 バイトまで圧縮します。

ネクスト・ヘッダが UDP トランスポート・ヘッダである場合、6LoWPAN は効率的なヘッダ圧縮を実現します。最も圧縮可能なポートを使用した場合、追加されるのはわずか 1 バイトになります。ただし、TCP トランスポート・ヘッダや ICMPv6 インターネット・ヘッダなど、その他のインターネット・プロトコル・スイート・ヘッダによる利点は提供しません。



13.1 ルーティング

メッシュ・ネットワークには主に 2 つのルーティング方式があり、一般に「ルートオーバー」と「メッシュアンダー」と呼ばれています。

13.1.1 ルートオーバー

ルートオーバー・アプローチでは、メッシュを IP ネットワーク（LAN やインターネット全体など）が拡張されたものとして取り扱います。この場合、すべてのノードが何らかの IP ルータとなり、以下の影響があります。

- ノードは、ICMPv6 隣接モート招待／通知メッセージおよびルータ招待／通知メッセージを送信することで、ネットワーク内の位置を検出する必要があります。どちらのメッセージも既存の規格（考えられるソリューションとして RFC7400 を参照）では効率的に圧縮されません。
- ノードでルーティング・テーブルを維持する必要があります。維持しない場合、有向非巡回グラフのルート（DODAG ルート）と呼ばれる入り口／出口ポイントから各ノードまでの距離に関連付けられたコスト関数を作成することで、単純な転送スキーマが使用されます。
- ノードはそれぞれのコスト関数を隣接モートに通知する必要があります。
- IP 層の分割サポートが必要です。
- メッシュ・ネットワークへの参加と LAN への参加が結び付けられているので、セキュリティ・スイートとその実装コストに影響があります。

このアプローチでは、メッシュが「インターネット」の一部であるかのように見えますが、実際のところ、標準的なイーサネットや Wi-Fi リンクとメッシュは根本的に性質（速度、帯域幅）が異なります。

13.1.2 メッシュアンダー

メッシュアンダー・アプローチでは、メッシュ内のルーティングと IP ネットワーク内のルーティングとは切り離して扱うので、事実上メッシュが 1 つの IP ホップに平坦化されます。つまり、パケットの送信元と宛先のみで IP ルーティングが行われることになり、以下の影響があります。

- メッシュ配信するには、追加のルーティング・メカニズム（ヘッダなど）が必要です。
- 隣接モートの検出で、高度に圧縮可能な UDP パケットを使用できます。
- メッシュ内を流れるパケットにはメッシュ・アドレスが予め関連付けられているので、大部分のアドレスを省略できます。
- ノードで IP ルーティング・テーブルを維持する必要はありません。
- IP 層の分割は必要ない場合があります。
- メッシュへの参加と LAN への参加は切り離されています。

このアプローチでは、メッシュとインターネットまたは LAN の間に低消費電力境界ルータ（LBR）が必要です。LBR は、トラフィックを調整することで、メッシュと IP ネットワークの帯域幅の違いに対応することができます。

13.2 SmartMesh IP の 6LoWPAN オプション

SmartMesh IP 組込みマネージャは、**sendData** と **sendIP** という 2 種類の API を使用してモートにデータを送信します。どちらの API もパケットの宛先として MAC アドレスを使用しますが、SmartMesh IP はステートレス・アドレスの自動構成を使用するので、メッシュ内でプレフィックスは必要ありません。**sendData** API には送信元ポートと宛先ポートが引数として渡され、マネージャが 6LoWPAN ヘッダを（大部分の 6LoWPAN フィールドは省略した状態で）構築し、モートに UDP パケットを送信します。**sendIP** API の場合、API クライアントで 6LoWPAN ヘッダを構築する必要があるので、より柔軟性が高くなりますが、現在の SmartMesh IP スタックでは、使用可能な符号化のうちの一部しか使用されていません。

同様に、VManager でモートにパケットを送信する場合も、**POST /motes/m/{mac}/dataPacket** コマンドと **POST /motes/m/{mac}/ipPacket** コマンドがあり、**dataPacket** ではマネージャが 6LoWPAN ヘッダを構築し、**ipPacket** ではアプリケーションが LoWPAN ヘッダを構築します。

6LoWPAN ヘッダには、IPv6 ヘッダの圧縮／解凍方法に対する指示が含まれます。

0 バイト				1 バイト					
011	TF	NH	HL	CID	SAC	SAM	M	DAC	DAM

6LoWPAN ヘッダ

フィールド	幅（ビット）	説明	指定できる値
011	3	固定フィールドで、予約済みのディスパッチ・タイプとして識別します。	b011
TF	2	トラフィック・クラスとフロー・ラベル - このスタックはトラフィック・クラスやフィールド・ラベルを伝送しません。	b11 =「トラフィック・クラスとフロー・ラベルは省略されている」
NH	1	ネクスト・ヘッダ	b1 =「ネクスト・ヘッダ・フィールドは LOWPAN_NHC を使用して圧縮および符号化されている」
HL	2	ホップ・リミット（IP ホップ - メッシュは 1 ホップとしてカウント）	b10 =「ホップ・リミット・フィールドは省略されており、ホップ・リミットは 64」
CID	1	コンテキスト識別子 - メッシュに含まれるトラフィック（送信元と宛先がどちらもモートまたはマネージャ）の場合、コンテキスト識別子 = 0。その他すべてのトラフィックの場合、送信元（LBR またはモート）がコンテキストを割り当て、コンテキスト識別子 = 1。1 バイトのコンテキスト識別子拡張（CIE）バイトがヘッダに付加されます。上位ニブルが送信元 ID で、下位ニブルが宛先 ID です。	b0 =「8 ビットの追加コンテキスト識別子拡張は使用しない」、または b1 =「8 ビットの追加コンテキスト識別子拡張を使用しており、パケット内に含む」

SAC	1	送信元アドレスの圧縮 - 特定の SAC および SAM の組み合わせのみ指定できます。	b0 =「送信元アドレスの圧縮でステートレス圧縮を使用」、または b1 =「送信元アドレスの圧縮でコンテキスト・ベースのステートフル圧縮を使用」
SAM	2	送信元アドレス・モード	SAC = b0 で SAM = 00 の場合、「128 ビット。インラインでフル・アドレスを伝送」 SAC = b1 で SAM = 11 の場合、「0 ビット。コンテキスト情報（およびリンク層アドレス）によりアドレスを決定」
M	1	マルチキャスト圧縮 - マルチキャスト・グループはサポートされないため、マルチキャスト・アドレスを使用した場合は常にメッシュ・ブロードキャストになります。	b0 =「宛先アドレスはマルチキャスト・アドレスでない」、または b1 =「宛先アドレスはマルチキャスト・アドレスである」。b1 の場合、DAC=1 で DAM=1。つまり、マルチキャスト・アドレスがインラインで伝送されることはありません。
DAC	1	宛先アドレスの圧縮 - 特定の DAC および DAM の組み合わせのみ指定できます。	b0 =「宛先アドレスの圧縮でステートレス圧縮を使用」、または b1 =「宛先アドレスの圧縮でコンテキスト・ベースのステートフル圧縮を使用」
DAM	2	宛先アドレス・モード	DAC = b0 で DAM = 00 の場合、「128 ビット。インラインでフル・アドレスを伝送」 DAC = b1 で DAM = 11 の場合、「0 ビット。コンテキスト情報（およびリンク層アドレス）によりアドレスを決定」

まとめ

メッシュ内トラフィック（送信元および宛先の両方ともコンテキスト = 0）の場合、6LoWPAN_IPHC ヘッダが、b011.11.1.10.0.1.11.0.1.11 = 0x7E77 となり、コンテキスト ID バイトは省略されます。

13.2.1 ネクスト・ヘッダの符号化

ネクスト・ヘッダの符号化が使用されるのは、分割、ICMPv6、TCP パケットなどで拡張ヘッダが使用される場合です。現在、SmartMesh IP スタックは UDP ネクスト・ヘッダのみをサポートしています。

13.2.2 UDP ヘッダの圧縮

UDP ヘッダは固有の圧縮形式である UDP LOWPAN_NHC を持ち、NHC 形式の代わりにこれを使用します。UDP LOWPAN_NHC は 1 バイトで、形式は以下のとおりです。

フィールド	幅(ビット)	説明	指定できる値
11110	5	固定フィールドで、UDP LOWPAN_NHC として識別します。	b11110
C	1	チェックサム - メッシュのセキュリティによって確実に破損なく配信されるので、UDP チェックサムは常に省略できます。	b1 =「2 バイトのチェックサムを省略」
P	2	ポート圧縮 - ユーザ・トラフィックに使用できるポートの制限はありませんが、アナログ・デバイセズのサンプル・アプリケーションは通常、0xF0Bx ポートを使用しています。	b00:送信元ポートと宛先ポートの両方に対して、16 ビットすべてがインラインで伝送されています。これは、両方のポートが圧縮可能な範囲外にある場合です。 b01:送信元ポートの 16 ビットすべてがインラインで伝送されています。宛先ポートの最初の 8 ビットは 0xf0 で、省略されています。宛先ポートの残りの 8 ビットはインラインで伝送されています。 b10:送信元ポートの最初の 8 ビットは 0xf0 で、省略されています。送信元ポートの残りの 8 ビットはインラインで伝送されています。宛先ポートの 16 ビットすべてがインラインで伝送されています。 b11:送信元ポートおよび宛先ポートの最初の 12 ビットは 0xF0B で、省略されています。それぞれの残りの 4 ビットはインラインで伝送されています。

上記に続く UDP 送信元／宛先ポートリストは、未圧縮(2B 送信元 + 2B 宛先)、部分圧縮(2B 未圧縮送信元／宛先 + 1B 圧縮(0xF0nn)送信元／宛先)、または完全圧縮(1B(0xF0Bn)送信元／宛先)のいずれかです。メッシュ制御トラフィックは UDP ポート 0xF0B0 を使用します。アプリケーション・トラフィック用にこのポートを使用することはできません。

まとめ

ポートが 0xF0Bx の範囲内にある場合、圧縮された UDP LOWPAN_NHC ヘッダは、b11110.1.11 = 0xF7 となり、ヘッダ自体は 0xsd となります(s = 送信元ポート、d = 宛先ポート)。

14 アプリケーション・ノート：VManager の OTAP アプリケーションの構築

モートのファームウェアはネットワーク上でライブ・アップグレードすることができ、この処理は、無線プログラミング（OTAP）と呼ばれます。このプロセスはファームウェア・イメージを安全な方法で 1 つ以上のモート上にダウンロードし、既存のファームウェアを新しいバージョンで置き換えます。低速な処理ですが、更新の最後にデバイスをリセットするまでは、通常のモート機能を中断しません。

アップグレード・プロセスの管理を担当するのは外部の OTAP アプリケーションで、どのモートを更新するか、どのソフトウェアをアップグレードするか、どの進捗情報を表示するか、などは OTAP アプリケーションの開発者が決めることができます。SmartMesh SDK では、OTAP の実行に使用できる Python リファレンス・アプリケーション（`VMgr_OTAPCommunicator.py`）を提供していますが、別の言語でカスタム・アプリケーションを作成することも可能です。このリファレンス・アプリケーションの詳細は、OTAP Communicator Documentation を参照してください。

アナログ・デバイセズは、`.otap2` ファイル形式のファームウェア更新用ソフトウェアを提供しています（LT5800 ベースのデバイス向け）。また、On-Chip Software Development Kit を使用して構築したカスタム・アプリケーション向けに、`.otap2` ファイルを生成するための `OTAPfile.py` を提供しています。

14.1 OTAP の仕組み

OTAP アプリケーション・フローには以下のステップがあります。

1. OTAP の対象とするモート（「受信リスト」）を決定します。
2. 更新対象の各モートに OTAP ハンドシェイクを送信して、OTAP プロセスを開始します。
3. OTAP ファイルを、パケットに収まる複数のブロックに分割します。
4. 分割した OTAP ファイルをネットワーク内のすべてのモートに送信します（ハンドシェイクを完了したモートのみがこの更新を使用します）。
5. すべてのモートの OTAP ステータスを監視します。
6. すべてのモートが完全なイメージを受信したら、ファームウェアをフラッシュにコミットします。

OTAP アプリケーションは、マネージャの API コマンド **POST /motes/m/{mac}/dataPacket**（「**dataPacket**」API）を使用して、OTAP プロトコル・パケットをモートに送信します。ここでは、ペイロード・フィールドが OTAP コマンドになります（以下で説明）。OTAP アプリケーションは、モートからの OTAP コマンド応答を使用して進捗を監視します。OTAP 応答を確認するため、OTAP アプリケーションはデータ通知をサブスクライブする必要があります（**GET /notifications**）。

API コマンド **dataPacket** は、モートの MAC アドレスをパラメータとして受け取ります。FF-FF-FF-FF-FF-FF-FF-FF を使用すると、ブロードキャスト可能なコマンド（OTAP データなど）をすべてのモートにブロードキャストできます。OTAP コマンドの送信元／宛先ポート = 0xF0B1 = 61617 です。送信 API に渡されるペイロードは、API ヘッダと API ペイロードで構成されます。コンテンツの詳細は下記を参照してください。

POST /motes/m/{mac}/dataPacket コマンドの詳細については、SmartMesh IP VManager API Guide を参照してください。

14.1.1 OTAP 処理の対象モートの「受信リスト」の準備

OTAP アプリケーションはモートの MAC アドレス・リストを入力として受け取るか、または各モートのバージョンを問い合わせて、アップグレード対象となるかどうかを確認することができます。モートへ問い合わせるため、アプリケーションで次の処理を実行する必要があります。

1. **GET /motes** コマンドを実行してモートのリストを構築します
2. リスト内の各モートに対して **GET /motes/m/{mac}/info** コマンドを使用し、**appSWRev** 文字列を取得します。アップグレード対象のバージョンを持つデバイスのみが「受信リスト」に追加されます。

API からは **appId** も返されますが、これは現在予約済みフィールドになっているので、カスタム・ファームウェアの識別には使用できません。

14.1.2 受信リストに含まれるモートとのユニキャスト・ハンドシェイク

OTAP アプリケーションは受信リスト内の各モートに対して、API コマンド **dataPacket** を使用し、34（0x22）バイトの **OTAPHandshake**（0x16）コマンドを含むパケットを送信します。

OTAPHandshake 要求

パラメータ	タイプ	説明
otapFlags	INT8U	OTAP フラグ = 0x07。この場合、ファイルが .otap2 ファイルであることと、以前の OTAP ファイルが存在する場合は上書きし、（削除できるように）一時マークをファイルに付けることがモートに伝えられます。
otapMIC	INT32U	ファイル・ペイロード用の特別な OTAP MIC で、.otap2 ファイルの MIC ではありません。下記の「otapMIC の計算」を参照してください。
fileSize	INT32U	ファイル・サイズは .otap2 ファイルのバイト数です。.otap2 ファイルのオフセット 0x4 に、4 バイトのサイズ・フィールドがあります。fileSize は、このフィールドの値に 8 バイトを足した値です。
blkSize	INT8U	ブロック・サイズは、各 OTAP データ・パケットに含まれるペイロードのバイト数であり、0x48（72 バイト、下記を参照）です。
fileInfo	dn_api_otap_fileinfo_t	OTAP ファイルの情報構造体です。下記を参照してください。

OTAP ファイル情報

OTAP ファイル情報の構造体の開始位置は、.otap2 ファイルのオフセット 0x8 です。これは、次の段階でモートが受信する .otap2 ファイルに対して検証されます。

dn_api_otap_fileinfo_t の構造

パラメータ	タイプ	説明
partition id	INT8U	パーティション ID = 0x02
flags	INT8U	ファイル・フラグ（ビットマップ）: 0x03 または 0x07 のどちらか ● b0 = 1（実行可能データ） ● b1 = 1（LZSS 圧縮済み） ● b2 = 0（ベンダおよびアプリケーション ID の検証）または 1（検証をスキップ） ビット 2 は、.otap2 ファイルの生成時に使用された設定によって異なります。工場出荷時は b2 = 0 に設定されています。
fileSize	INT32U	未圧縮の.bin イメージのサイズ
exeStartAddr	INT32U	実行可能データの開始アドレス = 0x00041020
exeVersion	APP_VER	OTAP 処理の対象アプリケーションのバージョン - **getParameter** 応答で APP_VER の説明を参照してください。
exeDependsVersion	APP_VER	依存バージョン（処理を進めるには、モートがこのバージョン以降でなければなりません）
exeAppId	INT8U	アプリケーション ID = 0x01*
exeVendorId	INT16U	ベンダ ID = 0x0001*
exeHwId	INT8U	ハードウェア ID = 0x03

*ベンダ ID およびアプリケーション ID は現在、予約済みフィールドになっているため、表内に記載した値を設定する必要があります。

モートはハンドシェイクを受け入れるとアクノリッジを返します。アクノリッジには、API 応答コード、OTAP 応答コード、要求内で指定されたファイルの **otapMIC**、**genDelay** が含まれます。genDelay は、モートの電力目標を達成するためにパケット間に必要な遅延（ミリ秒単位）です。OTAP アプリケーションは、受信リスト内のすべてのモートから返された遅延のうちの最大値か、1 下りフレーム（組込みマネージャの場合デフォルトで約 2 秒）のどちらか大きい方を間隔としてパケットを送信する必要があります。応答コードが RC_OK ではない場合、応答コードのみが返されます。

OTAPHandshake 応答

パラメータ	タイプ	説明
rc	INT8U	API 応答コード
otapRc	INT8U	OTAP 固有のリターン・コード（下記を参照）
otapMIC	INT32U	**OTAPHandshake** 要求で指定された otapMIC
genDelay	INT32U	パケット間に要求される遅延（ミリ秒単位）

14.1.3 otapMIC の計算

128 ビットの AES CBC 出力を計算することで、ファイル（平文）全体に対する 4 バイトの **otapMIC** を算出できます。これは OTAP アプリケーション内または帯域外（openssl の使用など）で実行でき、OTAP アプリケーションへの入力としてモートのバイナリと共に提供できます。アプリケーションは、MIC 対象データに加えて、16 バイトの鍵と 16 バイトのノンス（初期化ベクトル）を入力として受け取ります。

- 鍵 = `c3 bd 8f 3c c7 c9 99 29 22 92 f3 f2 a2 9d c3 10`
- ノンス = `00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`

必要に応じてゼロをパディングすることで、このファイルを 16 バイトの倍数のブロック・サイズに拡張できます。

otapMIC は、チェーンに含まれる最後の出力ブロックの最初の 4 バイトです。

例

データ（27 バイト） = `81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90`

`71 72 73 74 75 76 77 78 79 7a 7b`

パディングされたデータ（32 バイト） = `81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90`

`71 72 73 74 75 76 77 78 79 7a 7b 00 00 00 00 00`

符号化されたデータ（32 バイト） = `4c f9 4f 26 a3 2d c2 5b 81 40 0b c5 30 d1 a7 e0`

`c5 a2 09 34 05 4b 7f b0 da 56 5d 89 53 c3 a8 fd`

otapMIC（4 バイト） = `c5 a2 09 34`

14.1.4 .otap2 ファイルからネットワーク・パケットへの分割

OTAP アプリケーションは`.otap2` ファイル全体をデータ・ブロックに分割して、下りブロードキャスト・パケットに収まるようにする必要があります。最大ブロック・サイズ（下記の **OTAPData** 要求内のペイロード・フィールド）は、72（0x48）バイトです。すべてのデータ・ブロックのサイズが **OTAPHandshake** に指定されたサイズと一致する必要がありますが、ブロック・サイズに合わせるために最後のブロックにパディングする必要はありません。

14.1.5 ネットワークへの OTAP データ・ブロックのブロードキャスト

OTAP は複数回の往復処理で構成されています。最初の往復処理では、OTAP アプリケーションが、78（0x4E）バイトの **OTAPData**（0x17）コマンドを含む **dataPacket** API を使用して、`.otap2` ファイル内の各データ・ブロックをブロードキャスト・アドレス（FF-FF-FF-FF-FF-FF-FF-FF）に送信します。モートはそれぞれの **OTAPData** パケットには応答しません。このファイルを処理するのは、受信リストに含まれ、ハンドシェイクを受け入れたデバイスのみです。

OTAPData 要求

パラメータ	タイプ	説明
otapMIC	INT32U	**OTAPHandshake** で指定した otapMIC と一致する必要があります。
block	INT16U	ブロック番号
payload	INT8U[]	データ・ブロック・ペイロード（72 バイト）

14.1.6 更新ステータスの監視

OTAP アプリケーションは、4（0x04）バイトの **OTAPStatus**（0x18）コマンドを含む **dataPacket** API を使用して、受信リスト内のすべてのモートにコマンドを送信し、まだ受信していないブロックを問い合わせることで、更新ステータスをチェックすることができます。その際、ネットワーク全体にブロードキャストすることも、受信リスト内の各モートに対して順番に直接送信（ユニキャスト）することもできます。どちらのケースでも、モートからの応答が失敗する場合、OTAP アプリケーションから **OTAPStatus** コマンドを再試行する必要があります。

OTAPStatus 要求

パラメータ	タイプ	説明
otapMIC	INT32U	**OTAPHandshake** で指定した otapMIC と一致する必要があります

モートから返されるアクノリッジには、API 応答コード、OTAP 応答コード、要求内で指定されたファイルの **otapMIC**、まだ受信していないブロック・リストが含まれます。応答コードが RC_OK ではない場合、応答コードのみが返されます。

OTAPStatus 応答

パラメータ	タイプ	説明
rc	INT8U	API 応答コード
otapRc	INT8U	OTAP 固有のリターン・コード（下記を参照）
otapMIC	INT32U	**OTAPStatus** 要求で指定された otapMIC
lostBlocks	INT16U[]	不足ブロックのリスト

受信リスト内のすべてのモートが **OTAPStatus** コマンドに応答したら、すべての不足データ・ブロックのリストが作成されます。このリストが次の往復処理で使用され、プロセスが繰り返されます。

モートが報告できる不足ブロックの最大数は 40 であり、これは総ブロック数の 1～2%です。通常、これは問題にはなりませんが、ネットワークの安定性が十分でない場合、最初の **OTAPStatus** チェックに基づいて、不完全な不足ブロック・リストがアプリケーションに作成される可能性があります。この場合、アプリケーションから既知のブロックで再試行してステータスをもう一度チェックするか（追加ブロックが返されます）、少数のブロックの送信後に定期的に

OTAP ステータスを取得することで、最初の往復処理の終了時点でネットワーク全体の不足ブロック・リストが完全になる可能性が高くなります。

受信リスト内のすべてのモートから、データ・ブロックをすべて受け取ったことが通知された時点で（通常これは何度か往復処理します）、モートに対してファイルをプログラムする準備が整います。

OTAP アプリケーションが進行情報を表示する方法はいくつかあります。以下にその例を挙げます。

- 現在の往復、送信済みブロック数、直前の往復終了時に受信報告されたブロック数（モート当たり、または平均値）を表示する
- ネットワークに対して完了した割合（%、平均値）を単純に表示する

14.1.7 受信リストへのコミット・コマンドのユニキャスト

ファームウェアをプログラムするには、4（0x4）バイトの **OTAPCommit**（0x19）コマンドを含む **dataPacket** API を使用します。OTAP アプリケーションは、受信リストに含まれ、すべてのデータ・ブロックを受信したと通知したモートに対してこのコマンドを送信します。このコマンドを受け取ると、モートはフラッシュを再プログラムして新しいファームウェアにアップグレードしたのちリセットします。

OTAPCommit 要求

パラメータ	タイプ	説明
otapMIC	INT32U	**OTAPHandshake** で指定した otapMIC と一致する必要があります

モートから返されるアクノリッジには、API 応答コード、OTAP 応答コード、要求内で指定されたファイルの **otapMIC**、コミット応答コードが含まれます。応答コードが RC_OK ではない場合、応答コードのみが返されます。

OTAPCommit 応答

パラメータ	タイプ	説明
rc	INT8U	API 応答コード
otapRc	INT8U	OTAP 応答コード（下記を参照）
otapMIC	INT32U	**OTAPCommit** 要求で指定された otapMIC

OTAPCommit 応答を送信した後、モートはリセットを実行してから再プログラムし、もう一度リセットした後で動作を再開します。

14.1.8 モートのリセットと再参加

OTAP 処理の完了後、OTAP アプリケーションはいくつかの方法で OTAP 結果を表示できます。その例を以下に示します。

- 受信リスト内のモートを含むテーブルを表示し、OTAP 後の再参加状況を示す
- このテーブルを拡張し、新しいファームウェア・バージョンを問い合わせてから表示する
- 成功メッセージを出力してから終了する

14.2 Q&A

- Q:すべてのブロックが受信される前にコミットを実行するとどうなりますか。
  - A:モートから DN_API_OTAP_RC_MIC エラーが返され、OTAP が中止されます。すべてのブロックを受信していないモートで、OTAP を再開する必要があります。OTAP アプリケーションが一定回数の往復処理後にタイムアウトするような設計になっている場合、このような状況が起きる可能性があります。

- Q:OTAP 処理中にモートがリセットするとどうなりますか。
  - A:該当モートでの OTAP 処理が中止されます。そのモートで OTAP を再起動する必要があります。例えば、500 ブロックを送信した後でモートがリセットした場合、OTAP アプリケーションは **OTAPHandshake** をモートに送信し、続いて **OTAPData** メッセージを送信することができます。ただし、その後の往復処理で最初のブロックを繰り返す必要があります。

- Q:`.otap2` ファイルの検証に失敗した(例:依存バージョンに適合しない)場合はどうなりますか。
  - A:**OTAPCommit** 中に、モートからエラーが返されます。

- Q:モートがコマンドに応答しない場合はどうなりますか。
  - A:ルーティングがタイムアウトするまで(約数分)、マネージャは、**OTAPHandshake** コマンドと **OTAPCommit** コマンドを再試行します。30 分以内に応答を受信しない場合、アプリケーションでこれらのコマンドを再試行しても問題ありません。モートが最初のコマンドを受信していたにもかかわらず、応答がマネージャに届いていなかった場合、モートが応答する可能性があります。**OTAPData**(場合によっては **OTAPStatus**)コマンドは 1 回だけブロードキャストされます。アプリケーションでこれらのコマンドを 2～3 回繰り返すと、ブロック配信の可能性が少し高くなりますが、プロセス速度が低下するので、通常は 1 回だけ再試行し、必要な場合はその後の往復処理で再送することを推奨します。

- Q:現在デバイス上にあるローダ(1.3.0.12 など)とは異なるローダ(1.0.5.4 など)を必要とするバージョンを OTAP 処理した場合はどうなりますか。
  - A:モートはファイル内の **otapMIC** を **OTAPHandshake** に含まれる MIC と比較するので、**OTAPCommit** は成功したように見えますが、デバイスの再起動時にローダは **otapMIC** を検証することができません。これは、ローダ 1.0.3 と 1.0.5 の間で、FIPS 140-2 に適合するためにシグネチャ・アルゴリズムが変更されているからです。Eterna Serial Programmer ソフトウェアを使用してローダ

を再プログラムする必要があります。モートの SPI プログラミング・インターフェースを使用して新しいファームウェアを受け入れられる場合、この処理は現場で実施できます。

- Q：OTAP 処理の速度を上げることはできますか。
  - A：ネットワークに流れるパケットの速度を上げるには、下りフレーム・サイズを変更する必要があります。これは組込みマネージャでは実行できませんが、VManager では可能です。受信リスト内の失敗したモートのみに **OTAPCommit** を実行し、OTAP を再起動するよりも、数ブロックを受信していないモートが「追いつく」のを待つ方が、通常は速くなります。

- Q：`OTAPFile.py` を使用して`.otap2` ファイルを生成し、OTAP アプリケーションによるプログラムをエラーなしで完了しましたが、モートではまだ古いバージョンが使用されています。なぜですか。
  - A：1.1.0.8 より前のバージョンのオンチップ SDK に含まれていた `OTAPFile.py` ファイルは、**otapMIC** の生成で正しくないシグネチャ・アルゴリズムを使用します。新しいバージョンにアップグレードしてください。

14.3 OTAP 応答コード

OTAP コマンドで返される応答コードは以下のとおりです。

名前	値	説明
DN_API_OTAP_RC_OK	0	コマンドが受け入れられました。
DN_API_OTAP_RC_LOWBATT	1	バッテリ電圧が低すぎてフラッシュ書込みができません。
DN_API_OTAP_RC_FILE	2	ファイル・サイズ、ブロック・サイズ、開始アドレス、実行サイズのいずれかが正しくありません。
DN_API_OTAP_RC_INVALID_PARTITION	3	無効なパーティション情報（ID、ファイル・サイズ）
DN_API_OTAP_RC_INVALID_APP_ID	4	アプリケーション ID が正しくありません。
DN_API_OTAP_RC_INVALID_VER	5	ソフトウェア・バージョンが OTAP に適合しません。
DN_API_OTAP_RC_INVALID_VENDOR_ID	6	無効なベンダ ID
DN_API_OTAP_RC_RCV_ERROR	7	
DN_API_OTAP_RC_FLASH	8	その他のフラッシュ・エラー
DN_API_OTAP_RC_MIC	9	アップロードされたデータに対する MIC の失敗
DN_API_OTAP_RC_NOT_IN_OTAP	10	OTAP ハンドシェイクが開始されていません。
DN_API_OTAP_RC_IOERR	11	I/O エラー
DN_API_OTAP_RC_CREATE	12	OTAP ファイルを作成できません。
DN_API_OTAP_RC_INVALID_EXEPAR_HDR	13	exe パーティション・ヘッダ・フィールド（「signature」または「upgrade」）の値が正しくありません。
DN_API_OTAP_RC_RAM	14	メモリを割り当てできません。
DN_API_OTAP_RC_UNCOMPRESS	15	解凍エラー
DN_API_OTAP_IN_PROGRESS	16	OTAP が実行中です。
DN_API_OTAP_LOCK	17	OTAP がロックアウトされています。

15 アプリケーション・ノート：組込みマネージャの OTAP アプリケーションの構築

モートのファームウェアはネットワーク上でライブ・アップグレードすることができ、この処理は、無線プログラミング（OTAP）と呼ばれます。このプロセスはファームウェア・イメージを安全な方法で 1 つ以上のモート上にダウンロードし、既存のファームウェアを新しいバージョンで置き換えます。低速な処理ですが、更新の最後にデバイスをリセットするまでは、通常のモート機能を中断しません。

アップグレード・プロセスの管理を担当するのは外部の OTAP アプリケーションで、どのモートを更新するか、どのソフトウェアをアップグレードするか、どの進捗情報を表示するか、などは OTAP アプリケーションの開発者が決めることができます。SmartMesh SDK では、OTAP の実行に使用できる Python リファレンス・アプリケーション（`OTAPCommunicator.py`）を提供していますが、別の言語でカスタム・アプリケーションを作成することも可能です。このリファレンス・アプリケーションの詳細は、OTAP Communicator Documentation を参照してください。

アナログ・デバイセズは、`.otap2` ファイル形式のファームウェア更新用ソフトウェアを提供しています（LT5800 ベースのデバイス向け）。また、On-Chip Software Development Kit を使用して構築したカスタム・アプリケーション向けに、`.otap2` ファイルを生成するための `OTAPfile.py` を提供しています。

15.1 OTAP の仕組み

OTAP アプリケーション・フローには以下のステップがあります。

1. OTAP の対象とするモート（「受信リスト」）を決定します。
2. 更新対象の各モートに OTAP ハンドシェイクを送信して、OTAP プロセスを開始します。
3. OTAP ファイルを、パケットに収まる複数のブロックに分割します。
4. 分割した OTAP ファイルをネットワーク内のすべてのモートに送信します（ハンドシェイクを完了したモートのみがこの更新を使用します）。
5. すべてのモートの OTAP ステータスを監視します。
6. すべてのモートが完全なイメージを受信したら、ファームウェアをフラッシュにコミットします。

OTAP アプリケーションは、マネージャの API コマンド **sendData** を使用して、OTAP プロトコル・パケットをモートに送信します。ここでは、ペイロードが OTAP コマンドになります（以下で説明）。OTAP アプリケーションは、モートからの OTAP コマンド応答を使用して進捗を監視します。OTAP 応答を確認するため、OTAP アプリケーションはデータ通知をサブスクライブする必要があります。

SendData 要求

パラメータ	タイプ	一覧	説明
macAddress	MAC_ADDR		宛先モートの MAC アドレス。すべてのモートにブロードキャストする場合、FF-FF-FF-FF-FF-FF-FF-FF を使用できます。
priority	INT8U	パケット優先順位	パケットの優先順位。OTAP では低の使用を推奨します。

srcPort	INT16U		送信元ポート
dstPort	INT16U		宛先ポート
options	INT8U		options フィールドは将来的な使用のために予約されています。0 を設定してください。
payload	INT8U[]		API ヘッダと API ペイロード（下記を参照）

sendData の詳細については、SmartMesh IP Embedded Manager API Guide を参照してください。**getParameter** などの通常の API コマンドの送信元／宛先ポート = 0xF0B8 = 61624 です。OTAP コマンドの送信元／宛先ポート = 0xF0B1 = 61617 です。

15.1.1 OTAP 処理の対象モートの「受信リスト」の準備

OTAP アプリケーションはモートの MAC アドレス・リストを入力として受け取るか、またはすべてのモートのバージョンを問い合わせて、アップグレード対象となるかどうかを確認することができます。モートに問い合わせを行うには、アプリケーションで次の処理を実行する必要があります。

1. API コマンド **getMoteConfig** を使用し、MAC アドレスにすべて 00 を指定し、**next** パラメータに true を設定します。次に、返された MAC アドレスを使用して、**getMoteConfig** から RC_END_OF_LIST が返されるまでプロセスを繰り返すことで、モートのリストを作成します。
2. リスト上の各モートに対して **sendData** コマンドを使用し、モートの API コマンド **getParameter<appInfo>**をリモート実行します。リモート・コマンドは修正されたバージョンの API ヘッダを使用するので、HDLC 符号化されておらず、フラグ・バイトもありません、

API コマンド

API ヘッダ	API ペイロード
コマンド + 長さ	応答コード（応答のみ） + メッセージ・ペイロード

getParameter コマンドの場合、API ヘッダは以下のようになります。

sendData の API ヘッダ

コマンド	長さ
02（getParameter）	1

API ペイロードが **getParameter** 要求の本体となり、パラメータ ID として **appInfo** を含みます。

getParameter 要求

パラメータ	タイプ	説明
paramId	INT8U	パラメータ ID = appInfo = 0x1E

以上から、**sendData** 要求のペイロード・フィールドは、`02 01 1E` となります。

モートは、モートに関する以下の情報を、API ヘッダ（ここでは `02 0a`）に続く 10 バイトの API 応答ペイロードに入れて返します。応答コードが RC_OK ではない場合、応答コードのみが返されます。

getParameter 応答

パラメータ	タイプ	説明
rc	INT8U	レスポンス・コード
paramId	INT8U	パラメータ ID = appInfo = 0x1E
vendorId	INT16U	ベンダ ID = 0001*
appId	INT8U	アプリケーション ID = 01*
appVer	APP_VER	アプリケーションのバージョン。シリアライズされた形式は以下のとおりです。 • INT8U - major - メジャー・バージョン • INT8U - minor - マイナー・バージョン • INT8U - patch - パッチ・バージョン • INT16U - build - ビルド・バージョン

*ベンダ ID およびアプリケーション ID は現在、予約済みフィールドになっています。将来のバージョンでは、カスタム・ファームウェアを識別するために使用する可能性があります。

API コマンド **getParameter** と列挙タイプの定義については、SmartMesh IP Mote Serial API Guide を参照してください。

15.1.2 受信リストに含まれるモートとのユニキャスト・ハンドシェイク

OTAP アプリケーションは受信リスト内の各モートに対して、34（0x22）バイトの **OTAPHandshake**（0x16）コマンドを含む **sendData** コマンドを実行します。

OTAPHandshake 要求

パラメータ	タイプ	説明
otapFlags	INT8U	OTAP フラグ = 0x07。この場合、ファイルが .otap2 ファイルであることと、以前の OTAP ファイルが存在する場合は上書きし、（削除できるように）一時マークをファイルに付けることがモートに伝えられます。
otapMIC	INT32U	ファイル・ペイロード用の特別な OTAP MIC で、.otap2 ファイルの MIC ではありません。下記の「otapMIC の計算」を参照してください。
fileSize	INT32U	ファイル・サイズは .otap2 ファイルのバイト数です。.otap2 ファイルのオフセット 0x4 に、4 バイトのサイズ・フィールドがあります。fileSize は、このフィールドの値に 8 バイトを足した値です。
blkSize	INT8U	ブロック・サイズは、各 OTAP データ・パケットに含まれるペイロードのバイト数であり、0x48（72 バイト、下記を参照）です。
fileInfo	dn_api_otap_fileinfo_t	OTAP ファイルの情報構造体です。下記を参照してください。

OTAP ファイル情報

OTAP ファイル情報の構造体の開始位置は、.otap2 ファイルのオフセット 0x8 です。これは、次の段階でモートが受信する .otap2 ファイルに対して検証されます。

dn_api_otap_fileinfo_t の構造

パラメータ	タイプ	説明
partition id	INT8U	パーティション ID = 0x02
flags	INT8U	ファイル・フラグ（ビットマップ）: 0x03 または 0x07 のどちらか ● b0 = 1（実行可能データ） ● b1 = 1（LZSS 圧縮済み） ● b2 = 0（ベンダおよびアプリケーション ID の検証）または 1（検証をスキップ） ビット 2 は、.otap2 ファイルの生成時に使用された設定によって異なります。工場出荷時は b2 = 0 に設定されています。
fileSize	INT32U	未圧縮の.bin イメージのサイズ
exeStartAddr	INT32U	実行可能データの開始アドレス = 0x00041020
exeVersion	APP_VER	OTAP 処理の対象アプリケーションのバージョン - 「**getParameter** 応答」で APP_VER の説明を参照してください。
exeDependsVersion	APP_VER	依存バージョン（処理を進めるには、モートがこのバージョン以降でなければなりません）

exeAppId	INT8U	アプリケーション ID = 0x01*
exeVendorId	INT16U	ベンダ ID = 0x0001*
exeHwId	INT8U	ハードウェア ID = 0x03

*ベンダ ID およびアプリケーション ID は現在、予約済みフィールドになっています。将来のバージョンでは、カスタム・ファームウェアを識別するために使用する可能性があります。

モートはハンドシェイクを受け入れるとアクノリッジを返します。アクノリッジには、API 応答コード、OTAP 応答コード、要求内で指定されたファイルの **otapMIC**、**genDelay** が含まれます。genDelay は、モートの電力目標を達成するためにパケット間に必要な遅延(ミリ秒単位)です。OTAP アプリケーションは、受信リスト内のすべてのモートから返された遅延のうちの最大値か、1 下りフレーム(組込みマネージャの場合デフォルトで約 2 秒)のどちらか大きい方を間隔としてパケットを送信する必要があります。応答コードが RC_OK ではない場合、応答コードのみが返されます。

OTAPHandshake 応答

パラメータ	タイプ	説明
rc	INT8U	API 応答コード
otapRc	INT8U	OTAP 固有のリターン・コード(下記を参照)
otapMIC	INT32U	**OTAPHandshake** 要求で指定された otapMIC
genDelay	INT32U	パケット間に要求される遅延(ミリ秒単位)

15.1.3 otapMIC の計算

128 ビットの AES CBC 出力を計算することで、ファイル(平文)全体に対する 4 バイトの otapMIC を算出できます。これは OTAP アプリケーション内または帯域外(openssl の使用など)で実行でき、OTAP アプリケーションへの入力としてモートのバイナリと共に提供できます。アプリケーションは、MIC 対象データに加えて、16 バイトの鍵と 16 バイトのノンス(初期化ベクトル)を入力として受け取ります。

- 鍵 = `c3 bd 8f 3c c7 c9 99 29 22 92 f3 f2 a2 9d c3 10`
- ノンス = `00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`

必要に応じてゼロをパディングすることで、このファイルを 16 バイトの倍数のブロック・サイズに拡張できます。

otapMIC は、チェーンに含まれる最後の出力ブロックの最初の 4 バイトです。

例

データ(27 バイト) = `81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90`

`71 72 73 74 75 76 77 78 79 7a 7b`

パディングされたデータ(32 バイト) = `81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90`

```
71 72 73 74 75 76 77 78 79 7a 7b 00 00 00 00 00
```

符号化されたデータ(32 バイト) = `4c f9 4f 26 a3 2d c2 5b 81 40 0b c5 30 d1 a7 e0`

```
c5 a2 09 34 05 4b 7f b0 da 56 5d 89 53 c3 a8 fd
```

otapMIC(4 バイト) = `c5 a2 09 34`

15.1.4 .otap2 ファイルからネットワーク・パケットへの分割

OTAP アプリケーションは.`otap2` ファイル全体をデータ・ブロックに分割して、下りブロードキャスト・パケットに収まるようにする必要があります。最大ブロック・サイズ(下記の **OTAPData** 要求内のペイロード・フィールド)は、72(0x48)バイトです。すべてのデータ・ブロックのサイズが **OTAPHandshake** に指定されたサイズと一致する必要がありますが、ブロック・サイズに合わせるために最後のブロックにパディングする必要はありません。

15.1.5 ネットワークへの OTAP データ・ブロックのブロードキャスト

OTAP は複数回の往復処理で構成されています。最初の往復処理では、OTAP アプリケーションが、78(0x4E)バイトの **OTAPData** (0x17)コマンドを使用して、`.otap2` ファイル内の各データ・ブロックをブロードキャスト・アドレス(FF-FF-FF-FF-FF-FF-FF-FF)に送信します。モートはそれぞれの **OTAPData** パケットには応答しません。このファイルを処理するのは、受信リストに含まれ、ハンドシェイクを受け入れたデバイスのみです。

OTAPData 要求

パラメータ	タイプ	説明
otapMIC	INT32U	**OTAPHandshake** で指定した otapMIC と一致する必要があります
block	INT16U	ブロック番号
payload	INT8U[]	データ・ブロック・ペイロード(72 バイト)

15.1.6 更新ステータスの監視

OTAP アプリケーションは、4(0x04)バイトの **OTAPStatus** (0x18)コマンドを使用して、受信リスト内のすべてのモートにコマンドを送信し、まだ受信していないブロックを問い合わせることで、更新ステータスをチェックすることができます。その際、ネットワーク全体にブロードキャストすることも、受信リスト内の各モートに対して順番に直接送信(ユニキャスト)することもできます。どちらのケースでも、モートからの応答が失敗する場合、OTAP アプリケーションから OTAPStatus コマンドを再試行する必要があります。

OTAPStatus 要求

パラメータ	タイプ	説明
otapMIC	INT32U	**OTAPHandshake** で指定した otapMIC と一致する必要があります

モートから返されるアクノリッジには、API 応答コード、OTAP 応答コード、要求内で指定されたファイルの **otapMIC**、まだ受信していないブロック・リストが含まれます。応答コードが RC_OK ではない場合、応答コードのみが返されます。

OTAPStatus 応答

パラメータ	タイプ	説明
rc	INT8U	API 応答コード
otapRc	INT8U	OTAP 固有のリターン・コード（下記を参照）
otapMIC	INT32U	**OTAPStatus** 要求で指定された otapMIC
lostBlocks	INT16U[]	不足ブロックのリスト

受信リスト内のすべてのモートが **OTAPStatus** コマンドに応答したら、すべての不足データ・ブロックのリストが作成されます。このリストが次の往復処理で使用され、プロセスが繰り返されます。

モートが報告できる不足ブロックの最大数は 40 であり、これは総ブロック数の 1～2%です。通常、これは問題にはなりませんが、ネットワークの安定性が十分でない場合、最初の **OTAPStatus** チェックに基づいて、不完全な不足ブロック・リストがアプリケーションに作成される可能性があります。この場合、アプリケーションから既知のブロックで再試行してステータスをもう一度チェックするか（追加ブロックが返されます）、少数のブロックの送信後に定期的にOTAP ステータスを取得することで、最初の往復処理の終了時点でネットワーク全体の不足ブロック・リストが完全になる可能性が高くなります。

受信リスト内のすべてのモートから、データ・ブロックをすべて受け取ったことが通知された時点で（通常これは何度か往復処理します）、モートに対してファイルをプログラムする準備が整います。

OTAP アプリケーションが進行情報を表示する方法はいくつかあります。以下にその例を挙げます。

- 現在の往復、送信済みブロック数、直前の往復終了時に受信報告されたブロック数（モート当たり、または平均値）を表示する
- ネットワークに対して完了した割合（%、平均値）を単純に表示する

15.1.7 受信リストへのコミット・コマンドのユニキャスト

ファームウェアをプログラムするには、4（0x4）バイトの **OTAPCommit**（0x19）コマンドを使用します。OTAP アプリケーションは、受信リストに含まれ、すべてのデータ・ブロックを受信したと通知したモートに対してこのコマンドを送信します。このコマンドを受け取ると、モートはフラッシュを再プログラムして新しいファームウェアにアップグレードしたのちリセットします。

OTAPCommit 要求

パラメータ	タイプ	説明
otapMIC	INT32U	**OTAPHandshake** で指定した otapMIC と一致する必要があります

モートから返されるアクノリッジには、API 応答コード、OTAP 応答コード、要求内で指定されたファイルの **otapMIC**、コミット応答コードが含まれます。応答コードが RC_OK ではない場合、応答コードのみが返されます。

OTAPCommit 応答

パラメータ	タイプ	説明
rc	INT8U	API 応答コード
otapRc	INT8U	OTAP 応答コード（下記を参照）
otapMIC	INT32U	**OTAPCommit** 要求で指定された otapMIC

OTAPCommit 応答を送信した後、モートはリセットを実行してから再プログラムし、もう一度リセットした後で動作を再開します。

15.1.8 モートのリセットと再参加

OTAP 処理の完了後、OTAP アプリケーションはいくつかの方法で OTAP 結果を表示できます。その例を以下に示します。

- 受信リスト内のモートを含むテーブルを表示し、OTAP 後の再参加状況を示す
- このテーブルを拡張し、新しいファームウェア・バージョンを問い合わせてから表示する
- 成功メッセージを出力してから終了する

15.2 Q&A

- Q：すべてのブロックが受信される前にコミットを実行するとどうなりますか。
  - A：モートから DN_API_OTAP_RC_MIC エラーが返され、OTAP が中止されます。すべてのブロックを受信していないモートで、OTAP を再開する必要があります。OTAP アプリケーションが一定回数の往復処理後にタイムアウトするような設計になっている場合、このような状況が起きる可能性があります。

- Q：OTAP 処理中にモートがリセットするとどうなりますか。
  - A：該当モートでの OTAP 処理が中止されます。そのモートで OTAP を再起動する必要があります。例えば、500 ブロックを送信した後でモートがリセットした場合、OTAP アプリケーションは **OTAPHandshake** をモートに送信し、続いて **OTAPData** メッセージを送信することができます。ただし、その後の往復処理で最初のブロックを繰り返す必要があります。

- Q:.otap2 ファイルの検証に失敗した(例:依存バージョンに適合しない)場合はどうなりますか。
  - A:**OTAPCommit** 中に、モートからエラーが返されます。

- Q:モートがコマンドに応答しない場合はどうなりますか。
  - A:ルーティングがタイムアウトするまで(約数分)、マネージャは、**OTAPHandshake** コマンドと **OTAPCommit** コマンドを再試行します。30 分以内に応答を受信しない場合、アプリケーションでこれらのコマンドを再試行しても問題ありません。モートが最初のコマンドを受信していたにもかかわらず、応答がマネージャに届いていなかった場合、モートが応答する可能性があります。**OTAPData**(場合によっては **OTAPStatus**)コマンドは 1 回だけブロードキャストされます。アプリケーションでこれらのコマンドを 2〜3 回繰り返すと、ブロック配信の可能性が少し高くなりますが、プロセス速度が低下するので、通常は 1 回だけ再試行し、必要な場合はその後の往復処理で再送することを推奨します。

- Q:現在デバイス上にあるローダ(1.3.0.12 など)とは異なるローダ(1.0.5.4 など)を必要とするバージョンを OTAP 処理した場合はどうなりますか。
  - A:モートはファイル内の **otapMIC** を **OTAPHandshake** に含まれる MIC と比較するので、**OTAPCommit** は成功したように見えますが、デバイスの再起動時にローダは **otapMIC** を検証することができません。これは、1.0.3 と 1.0.5 の間で、FIPS 140-2 に適合するためにシグネチャ・アルゴリズムが変更されているからです。Eterna Serial Programmer ソフトウェアを使用してローダを再プログラムする必要があります。モートの SPI プログラミング・インターフェースを使用して新しいファームウェアを受け入れられる場合、この処理は現場で実施できます。

- Q:OTAP 処理の速度を上げることはできますか。
  - A:ネットワークに流れるパケットの速度を上げるには、下りフレーム・サイズを変更する必要があります。これは組込みマネージャでは実行できませんが、VManager では可能です。受信リスト内の失敗したモートのみに **OTAPCommit** を実行し、OTAP を再起動するよりも、数ブロックを受信していないモートが「追いつく」のを待つ方が、通常は速くなります。

- Q:`OTAPFile.py` を使用して.otap2 ファイルを生成し、OTAP アプリケーションによるプログラムをエラーなしで完了しましたが、モートではまだ古いバージョンが使用されています。なぜですか。
  - A:1.1.0.8 より前のバージョンのオンチップ SDK に含まれていた `OTAPFile.py` ファイルは、**otapMIC** の生成で正しくないシグネチャ・アルゴリズムを使用します。新しいバージョンにアップグレードしてください。

15.3 OTAP 応答コード

OTAP コマンドで返される応答コードは以下のとおりです。

名前	値	説明
DN_API_OTAP_RC_OK	0	コマンドが受け入れられました。
DN_API_OTAP_RC_LOWBATT	1	バッテリ電圧が低すぎてフラッシュ書き込みできません。
DN_API_OTAP_RC_FILE	2	ファイル・サイズ、ブロック・サイズ、開始アドレス、実行サイズのいずれかが正しくありません。
DN_API_OTAP_RC_INVALID_PARTITION	3	無効なパーティション情報（ID、ファイル・サイズ）
DN_API_OTAP_RC_INVALID_APP_ID	4	アプリケーション ID が正しくありません。
DN_API_OTAP_RC_INVALID_VER	5	ソフトウェア・バージョンが OTAP に適合しません。
DN_API_OTAP_RC_INVALID_VENDOR_ID	6	無効なベンダ ID
DN_API_OTAP_RC_RCV_ERROR	7	
DN_API_OTAP_RC_FLASH	8	その他のフラッシュ・エラー
DN_API_OTAP_RC_MIC	9	アップロードされたデータに対する MIC の失敗
DN_API_OTAP_RC_NOT_IN_OTAP	10	OTAP ハンドシェイクが開始されていません。
DN_API_OTAP_RC_IOERR	11	I/O エラー
DN_API_OTAP_RC_CREATE	12	OTAP ファイルを作成できません。
DN_API_OTAP_RC_INVALID_EXEPAR_HDR	13	exe パーティション・ヘッダ・フィールド（「signature」または「upgrade」）の値が正しくありません。
DN_API_OTAP_RC_RAM	14	メモリを割り当てできません。
DN_API_OTAP_RC_UNCOMPRESS	15	解凍エラー
DN_API_OTAP_IN_PROGRESS	16	OTAP が実行中です。
DN_API_OTAP_LOCK	17	OTAP がロックアウトされています。

16 アプリケーション・ノート：配置の計画

16.1 範囲の推定

ある距離でのデバイス相互の通信を良好にするには、ハードウェアの集積化をどう選択するかが影響し、中でもアンテナの選択が最も明らかに影響します。集積化後、デバイスの配置によって有効範囲が数桁変わることがあります。有効範囲の一方の端では、高所の柱や塔に設置され、ネットワーク内の他のモートへの見通し線に障害物がないデバイスの範囲は 1000m 以上です。もう一方の端では、地表や大きな金属物に隣接して置かれたデバイスの有効範囲は 10m 以下になることがあります。したがって、「貴社製品の無線の範囲はどのくらいか」と顧客から尋ねられた場合、ある意味その質問には意味がないか、答えることができません。伝送電力や受信感度、得られるリンク・バジェットについてはデータシートを参照できますが、顧客は顧客の製品の開発時と、顧客が予想した配置に似た実際の環境での評価時に行う選択によって範囲を決めます。

開発の初期にはデバイスが 50m の間隔で動作するという計画を顧客が立てることを推奨します。最初の数回の「標準的な」配置を分析すると、範囲の標準的な数値の増減を導き出すことができます。配置計画が要求することは、単純に既存の 3 つ以上のデバイスのこの範囲内に各モートを配置することです。信頼できるメッシュを形成するため、各デバイスには複数の隣接デバイスと、したがって非常に多くの接続機会が必要です。範囲内にある唯一の他デバイスと最大限の間隔で一列にモートを配置すると、モートがリセットされやすくデータが消失しやすい脆弱なネットワークになります。

16.2 配置の綿密な計画策定

環境での範囲が決まったら、縮尺図を使用して、ネットワークで求められるすべての検出点にモートを配置することができます。可能な場合は、モートの分布の中央付近に AP を配置して、待ち時間およびモートの消費電力を低減してください。AP の位置に印をつけます。上記で推定した範囲を 50m と想定して、マネージャを中心に半径 50m の円を描きます。この円内にあるすべてのモートが AP と直接通信できるとは限りませんが、円の外側にある一部のモートは直接通信できるので、平均すると釣り合います。この円の内側にあるモートの数は、この配置での 1 ホップ・モートの数に近い値です。

次に、AP を中心に半径 100m の円を描きます。50m と 100m の間のリングの中にあるモートの数は、2 ホップ・モートの数に近い値です。半径を延ばしてこの手順を繰り返し、すべてのモートが囲まれるまで続けて、各ホップにモートがいくつあるかをメモします。これらのホップ・カウントはすぐに使います。

確認することがまだ 2 つあります。

- AP を含む各モートは、他の 3 つのデバイスの推定範囲内に存在する必要があります。
- ネットワークは 8 ホップ以内である必要があります。これより深いネットワークは確かに可能でとにかく機能はしますが、モデル化するのが困難です。

16.3 電力と待ち時間の推定

Dust は、両製品ファミリのネットワーク性能を推定する SmartMesh Power and Performance Estimator を提供しています。このツールを使用することにより、所定のトポロジ、パケット・レート、ホップ当たりの待ち時間、および目的の存続期間に対応するバッテリ・サイズを推定できます。

入力項目は以下のとおりです。

- 各ホップでのモートの数
- 報告間隔およびパケット・サイズ
- ネットワーク構成
- 経路安定性

これらを入力することにより、Estimator（推定プログラム）は、まずネットワークが形成されるかどうかを評価し、形成される場合は以下の項目を示します。

- ホップごとのモートの平均電流
- ホップごとの平均待ち時間
- ネットワークの参加時間
- 参加しているモートの電流

顧客に電力量計算の検討を案内するという前提で、構成例の使用を推奨します。

⚠ 実際の配置で消費電力と待ち時間に影響する要因は多数存在し、配置した時点でのネットワークの経路安定性（経時変化あり）や接続性（深さ方向のホップ数）などがあります。Estimator は、所定のホップでのモートの妥当な平均電力を示します（平均の 3 倍のホップがあるモートが存在する場合があります）。これらの推定値は予想設定値に対応するものです。実際のネットワーク内性能は異なります。

17 アプリケーション・ノート：CLI を使用した組込みマネージャのネットワーク健全性の予測

配置済みで稼働中のネットワークの健全性を評価することは、長期的な性能目標が達成可能であることを確認するために重要です。ネットワーク健全性の検証はシンプルであり、組込みマネージャの CLI を通じてすぐに入手できる情報の解釈に基づいています。ネットワークがこのネットワーク健全性検証テストを不合格になった場合、通常は重要な位置にモートを追加することで問題が改善されます。

17.1 目的

ネットワークが配置されて稼働状態になったら、重要なのはネットワークが健全であることの検証であり、より重要なことは将来も健全であり続けることの検証です。ネットワークは必要なすべての診断データを収集します。このデータはマネージャのソフトウェア・インターフェースで提供されます。製品の開発の早い段階で、ソフトウェアをマネージャに統合する開発者は、診断が重要であることを認識していることが重要です。ネットワークの健全性検証を自動化するツールを作成することは、特に無線に関して懐疑的な脳裏に信頼感を植え付ける極めて良い投資となります。プログラムを使用した健全性監視については、「SmartMesh IP ネットワークの健全性のモニタ」アプリケーション・ノートを参照してください。

この評価を手動で実施することもできます。以下に説明する作業によって、ネットワークから撤退した方が安全（「ゴーサイン」）かどうかが分ります。またこれは、適切に設置されたネットワークに問題があるというまれな事例で問題の原因を特定するのに役立ちます。

17.2 概要

ネットワークの検証では、2 つの複合的な質問に答えることが必要です。

1. ネットワークは順調に見えますか？
2. ネットワークの構成要素は順調に機能していますか？

これら両方の質問に対して求められる答えは「はい」です。これら両方の質問に対する答えが「はい」の場合、テストは合格しており、該当のネットワークは当分の間順調に稼働すると予想されます。

17.3 ネットワークは順調に見えますか？

ネットワーク評価テストのこの部分では、ネットワークに関する非常に単純な 3 つの観測上の質問に答えることが必要です。質問は以下のとおりです。

1）データの信頼性は高いですか？

優れた配置では、ネットワークのデータ配信率が 100%近くになります。Dust のネットワークはデータを消失する仕組みがほとんどない方法で構築されているので、99.99%を超えるデータ信頼性が期待されます。`show stat` コマンドを使用すると、消失パケットと到着パケットの合計数が出力されます。1 - Lost/Arrived > 99.99%になることを確認してください。以下の例では、消失パケット数はゼロで到着パケット数は 8279 になっているので、信頼性は 100%です。

```
> show stat
Manager Statistics ---------------------------------
   established connections      : 1
   dropped connections          : 0
   transmit OK                  : 0
   transmit error               : 0
   transmit repeat              : 0
   receive  OK                  : 313
   receive  error               : 0
   acknowledge delay avrg       : 0 msec
   acknowledge delay max        : 0 msec
Network Statistics ---------------------------------
   reliability: 100% (Arrived/Lost: 8279/0)
   stability:    90% (Transmit/Fails: 22012/2175)
   latency:     100 msec
```

2）参加動作は正しいですか？

この質問の前半部分は、「デバイスはすべて参加しましたか？」です。配置されたデバイスの数を確実に知ることができるのは設置作業者だけです。設置作業者は 100 デバイスを外に置いた場合、100 デバイスが参加したことを確認する必要があります。この質問の後半部分は、すべてのデバイスが厳密に 1 回参加したことを確かめるものです。あるデバイスが 2 回以上参加した場合、そのデバイスのネットワーク内での活動時間は、デバイスが絶え間なく脱落して再参加しているわけではないと確信するのに十分なほど長時間ですか？ネットワークの構築中に脱落して 1 回再参加したデバイスは、構築完了後、長時間経過してからリセットされたデバイスほど問題ではありません。`sm` コマンドを使用すると、ネットワーク内のすべてのモートが MAC アドレスと共に表示され、各デバイスの再参加回数も表示されます。

```
> sm
   MAC                    MoteId  State  Nbrs   Links Joins  Age   StateTime
00-17-0D-00-00-30-0A-F4     1      Oper     6     114      1    0   0-01:53:26
00-17-0D-00-00-3F-FC-04     2      Oper     8      39      1  188   0-01:53:08
00-17-0D-00-00-38-16-2A     3      Oper     4      31      1  178   0-01:52:55
00-17-0D-00-00-3F-FC-18     4      Oper     3      58      1  167   0-01:52:47
00-17-0D-00-00-38-12-84     5      Oper     2      11      1  163   0-01:52:42
00-17-0D-00-00-3F-FD-3B     6      Oper     5      66      1  135   0-01:52:15
00-17-0D-00-00-38-16-1A     7      Oper     2      15      1  117   0-01:51:56
00-17-0D-00-00-38-13-21     8      Oper     3      19      1   87   0-01:51:26
00-17-0D-00-00-38-14-39     9      Oper     3      24      1   29   0-01:50:28
```

3)ネットワークはメッシュ型に見えますか？

ネットワーク内の各モートに対して `show mote` コマンドを実行し、メッシュ内すべてのモートに親が 2 つあることを確認します。以下の例で、モート 2 は 8 つの隣接モートを持ち、そのうちの 2 つが親になっています。1 ホップのリングの中には、親が 1 つだけのモートが厳密に 1 つだけ存在するはずです。それで OK であり、期待どおりです。`show mote *`という形式でこのコマンドを実行すると、ネットワーク内のすべてのモートが CLI に出力されます。

```
show mote 2
Mote #2, mac: 00-17-0D-00-00-3F-FC-04
   State:Oper, Hops: 1.2, Uptime:  0-01:58:42, Age: 222
   Regular. Route/TplgRoute.
   Power Cost: Max 65534, FullTx 110, FullRx 65
   Capacity links: 200, neighbours 31
   Number of neighbors (parents, descendants): 8 (2, 6)
```

17.4 ネットワークの構成要素は順調に機能していますか？

この部分のネットワーク評価テストでは、メッシュ内の接続性の詳細に関する 3 つの定量的な質問に答えることが必要です。3 つの質問は以下のとおりです。

1)1 ホップのリングの中には十分な数のモートがありますか？

ネットワーク内のすべてのトラフィックは、ネットワーク・マネージャに接続されているモートである AP モートに集まります。この単一モートは、送出されるすべてのデータにとって非常に重要です。また、AP と直接通信するすべての **1 ホップ**・モートも同様に重要です。メッシュ内で最も激しく動作しているモートは、1 ホップのリングの中にあります。これらはその子孫モートからのトラフィックを転送しているモートです。1 ホップ・モートの数が増えるほど、トラフィックのバランスを調整して、単一モートのリセットを切り抜ける機会が増えるので、ネットワークにとっては好都合です。1 モートを除去すると多くのモートのデータが失われるシステムは決して作りたくありません。おおよその目安として、1 ホップのリングの中には、少なくとも 5 モートまたは全体の 10%のいずれか多い方の数のモートが必要です。ネットワーク内に 120 モートが存在し、そのうち 1 ホップ・モートが 10 個である場合、このネットワークが崩壊すると保証されるわけではありません。ただし、専門家でなくとも 2 択（はい／いいえ）の質問に答えることができるように、答えが「いいえ」の場合に何をしたらよいかについてすぐに実施可能な手順を示し、定量化可能な優れたガイドラインがここには必要です。以下の例では、AP（モート ID = 1）に 6 つの隣接モートが存在します。AP は、その定義からして親を

持たないので、これらの隣接モートはすべて子モートです。このネットワーク例では、8 つのモートと 1 つの AP があるので、1 ホップ・モートが 6 つあれば 1 ホップ条件が満たされます。

```
Mote #1, mac: 00-17-0D-00-00-30-0A-F4
   State:Oper, Hops: 0.0, Uptime:  0-02:12:37, Age: 0
   Power. Route/TplgRoute.
   Power Cost: Max 65535, FullTx 0, FullRx 0
   Capacity links: 250, neighbours 99
   Number of neighbors (parents, descendants): 6 (0, 8)
```

2)すべてのモートには十分な数の良質な隣接モートがありますか？

この手順では、最後のモートが参加後 15 分経過するまで待機し、ネットワーク内の検出済み経路をすべて調べて、すべてのモートに良質の隣接モートが十分存在することを確認する必要があります。最低でも、すべてのモートに良質の隣接モートが 3 つ以上存在することが必要です。良質の隣接モートとは、このモートが-75dBm を超える性能で受信できるか、50%を超える経路安定性を備えた隣接モートのことです。これらの経路は必ずしも現在使用中である必要はなく、必要なのはネットワークによる検出と報告が行われることだけです。この詳細情報をチェックするには、`show mote -a` コマンドを使用します。このコマンドは未使用経路も表示するので、「Neighbors」という見出しのリストを確認します。リンク品質(Q)が 50%を上回る行数を数えます。以下の例では、モート 9 によって、ネットワーク内にあるその他の 8 つのモートがすべて検出され、「Neighbors」のセクションに 1 行ずつ表示されています。表示された Q の値はすべて 50%を超えているので、良好な隣接モートが 8 つあることになり、条件を満たします。

```
> show mote -a 9
Mote #9, mac: 00-17-0D-00-00-38-14-39
   State: Oper, Hops: 1.6, Uptime: 0-02:19:09, Age: 400
   Regular. Route/TplgRoute.
   Power Cost: Max 65534, FullTx 110, FullRx 65
   Capacity links: 200, neighbours 31
   Number of neighbors (parents, descendants) : 2 (2, 0)
   Bandwidth total / mote exist (requested)   : 864 / 864 (993)
      Links total / mote exist (requested)    : 7.0 / 7.0 (6.1)
      Link Utilization                        : 1.0
   Number of total TX links (exist / extra)   : 7 / 0
   Number of links   : 15
     Compressed      : 5
     Upstream tx/rx  : 7 (7/0) (Rx10=0.0)
     Downstream rx   : 3
   Neighbors:
      -> # 1 Q: 84% RSSI: -56/0
      -> # 2 Q: 94% RSSI: -32/-35
      -- # 3 Q: 74% RSSI: 0/0
      -- # 4 Q: 74% RSSI: 0/0
      -- # 5 Q: 74% RSSI: 0/0
      -- # 6 Q: 94% RSSI: 0/0
      -- # 7 Q: 74% RSSI: 0/0
      -- # 8 Q: 74% RSSI: 0/0
```

3)リンクの限界に達しているか限界に近いモートはありますか？

現在の全製品では、180 以上のリンクがあるモートはネットワークに帯域幅問題の危険性を示しています。これをチェックするには、もう一度 sm コマンドの出力を見て、「Links」列に注目します。この例では、AP のリンク数が 114 で、最もリンクの多いモートでもリンク数は 66 なので、すべてが問題なくリンク限界を下回っています。

```
> sm
     MAC                 MoteId  State  Nbrs   Links Joins  Age   StateTime
00-17-0D-00-00-30-0A-F4     1      Oper     6     114     1    0   0-01:53:26
00-17-0D-00-00-3F-FC-04     2      Oper     8      39     1  188   0-01:53:08
00-17-0D-00-00-38-16-2A     3      Oper     4      31     1  178   0-01:52:55
00-17-0D-00-00-3F-FC-18     4      Oper     3      58     1  167   0-01:52:47
00-17-0D-00-00-38-12-84     5      Oper     2      11     1  163   0-01:52:42
00-17-0D-00-00-3F-FD-3B     6      Oper     5      66     1  135   0-01:52:15
00-17-0D-00-00-38-16-1A     7      Oper     2      15     1  117   0-01:51:56
00-17-0D-00-00-38-13-21     8      Oper     3      19     1   87   0-01:51:26
00-17-0D-00-00-38-14-39     9      Oper     3      24     1   29   0-01:50:28
```

18 アプリケーション・ノート：よくある問題と解決策

18.1 概要

ネットワークを構築するときの目標は、信頼できるサービスを提供すると同時にワイヤレス・デバイスでの消費電力をできるだけ低く抑えることです。リンクはエネルギーを消費するので、ネットワーク存続期間の参加段階および定常状態段階では、モートに与えられるリンク数が、モートを通過する予想トラフィック量をできるだけ少ないリンクで十分伝送できるように設定されます。マネージャは、サービス要求を正確に報告するモートと、平均の安定性が 50%より高い各経路に依存します。ボトルネックがある場合、つまり消費電力またはメモリの制約によっていずれかのモートがリンクを使い果たすと、このモートの子孫モートからすべてのトラフィックを伝送するための帯域幅が不足する可能性があります。

性能が低下しているネットワークの症状を以下に示します。

- 形成時間が長いモートがリセットする
- パケットの待ち時間に大きなばらつきがある
- マネージャがいずれかのモートからの消失パケットを報告する

これらの性能問題は、通常は以下のいずれか 1 つ以上が原因です。

- 不十分な接続性 - 良質の経路を持つ隣接モートが不足している
- 干渉 - 帯域内 WiFi または Bluetooth が存在するか、または強力な帯域外干渉物が近くにあり、経路安定性が低下している
- 申し込み超過 - 許容数を超えたサービス要求をモートが報告しているため、輻輳が生じている

モートはその内部統計情報と経路統計情報を健全性レポート・パケットで報告します。これらの統計情報は 2 または 3 パケットに分割され、15 分おきに生成されます。特に、現在モートが使用中の経路について、報告された経路安定性の値を確認してください。モートはその内部パケット・キューの最大サイズと平均サイズを報告します。Dust のネットワークは、どの時点のどのモートにも複数のパケットが存在することはめったにないように準備されているので、キューの長さの平均値がゼロ以外の場合、通常は問題があることを示しています。

また、マネージャもネットワーク内のすべてのモートについて、ネットワークのトポロジとリンクの割り当てを常時監視しています。この観点から、リンクの限界に達しているモートや、連続番号の一部を飛ばしてパケット消失を示しているモートがあるかどうかをすぐに識別できます。更に、マネージャはモートがリセットしたらアラームを出します。

干渉も、多くの Dust ネットワークをまとめて配置している結果と考えて差し支えありません。現在の製品ラインでは、ネットワークは時間または帯域幅の割り当ての点で同期しないので、あるネットワークからの伝送が別のネットワークと同時に同じチャンネルで行われることがあります。このネットワーク間干渉によって深刻な性能問題が発生する可能性を低くするために対策を実施してきましたが、複数の共同設置ネットワーク環境では全体的な経路安定性が低めに見える可能性があり、経路安定性はトラフィックの全合算量の関数です。経路安定性が低くても、データ信頼性が必ずしも低くなるとは限らないので注意してください。データの信頼性が低くなるには、厳しい干渉が発生する必要があります。

18.2 参加モートなし

モートがマネージャにまったく参加しない理由は以下のとおりです。

- マネージャが動作していない。
- マネージャに AP モートが接続されていない。
- AP モートにアンテナが接続されていない。
- マネージャのネットワーク ID または参加鍵（あるいはその両方）がモートのセキュリティ資格情報と一致しない。
- マネージャのアクセス制御リストにモートが含まれていない。
- モートがすべて AP から遠く離れて配置されている。
- モートに電源が入っていない。
- モートのセンサー・ファームウェアが join コマンドを正しく送信していない。

18.3 まとまった数のモートが参加しない

参加するモートもあれば参加しないモートもある場合は、少なくともマネージャと AP が機能している状態を確立しています。一部のモートが参加しない理由として、以下のことが考えられます。

- 一部のモートの配置場所が離れすぎている。
- マネージャの最大モート数に達している。
- 一部のモートには、マネージャに参加するための正しいセキュリティ資格情報（ネットワーク ID、参加鍵、ACL 項目）がない。
- 範囲内のモートがリーフ・ノードとして構成されている。

18.4 1 つのモートが参加しない

参加に失敗するモートの数がネットワーク・サイズと比較して小さい（つまり 100 個中 1 個などの）場合、可能性のある理由は以下のとおりです。

- 該当のモートに RF の問題がある（モートが壊れているか、アンテナが取り付けられていないなど）。
- 該当のモートのセキュリティ資格情報に誤りがある。
- 該当のモートに電源が入っていない。
- モートの最大数に到達している。
- 該当のモートは残りのネットワークから遠く離れて配置されている。

18.5 1 つのモートによる離脱と再参加の繰り返し

モートはネットワークへの接続状態を無期限に維持する必要があります。1 つのモートが参加後ネットワークを離脱し、再び参加する理由として、以下のことが考えられます。

- モートに電源の問題があり、そのためモートがリセットされる。
- 隣接モートへの RF 接続に余裕がない。
- 隣接モートへの RF 接続が非常に過渡的で不安定である。
- （筐体内や大型の障害物の背後のように）RF 接続が切断されてから再確立できる場所にモートがある。

18.6 動作範囲内のデバイスの経路安定性が低い

これは干渉が原因と予想されるので、モートを互いに近づけて配置し、SNR を高くします。

18.7 中継器を設置する必要があるが、既にモートが最大数に達している

中継器に接続能力が必要だった場合：モートを 1 つ取り外して中継器を配置するか、モートを再配置して経路を短くします。

中継器に第 1 ホップの帯域幅が必要だった場合：報告率を低減するか、モートを外側から第 1 ホップのリングの中に移動します。

18.8 データ待ち時間が予想より長い

データ待ち時間を個々のデバイスで短くするには、要求のサービス期間を短くしながら発行率は未変更のままにします。ただし代償として（該当のモートとその先祖（上位）モートの）バッテリ寿命が短くなります。また、基本帯域幅を広げれば、ネットワーク全体での待ち時間を改善できます。

18.9 ネットワークが最適に見えない経路を使用している

ネットワークは消費電力が最低限で済むように絶えず最適化しようとします。その一環として、定期的に新しい経路が試行されます。経路安定性に加えて、効果を発揮し始めるその他の検討事項があります。

19 アプリケーション・ノート：プロビジョニング係数の変更によるマネージャ・スループットの増加

19.1 概要

マネージャは、**プロビジョニング**によって経路安定性の短期間の変化を監視し、モートが発生するトラフィックの関数としてモートにリンクを割り当てます。デフォルトでは、SmartMesh ネットワークのプロビジョニング係数は 3 倍です。モートを通過すると予想されるパケットごとに 3 つのリンクを獲得します。これにより、デバイスは安定性が一時的に33%まで低下しても乗り切ることが可能であり、そのときデータの消失につながるキュー満杯の危険性はありません。ただし、マネージャのアクセス・ポイントで使用可能なリンクの数は限られているので、プロビジョニングによってパケットのスループットには上限が定められます。

マネージャのスループットの上限があるアプリケーションでは、ネットワークを小規模なサブネットワークに分割して総スループットを高める方法を推奨します。この解決策では不十分な場合、顧客はプロビジョニング係数を変更して限度内でスループットを増やすことができます。Dust では、プロビジョニング係数を 1.5 倍より低い値には設定しないことを推奨します。経路安定性が 70%未満に低下することは、これまでに観測した顧客のネットワークでは珍しいことではないので、ネットワークが低ノイズ、低マルチパス環境で動作しているという確実な認識がない限り、プロビジョニング係数を低く設定するのは危険です。一般に、経路安定性が 67%より高い場合は、観測された最低の経路安定性の逆数にプロビジョニング係数を設定します。

SmartMesh マネージャは、データ・トラフィックの伝送以外の機能を対象にリンクを使用します。**例えば**、十分な再試行回数でキープアライブを送信して経路アラームを防止します。このため、一部のモートには、トラフィック単独で必要な数より多くアクセス・ポイントへのリンクが存在することがあります。アクセス・ポイントがリンクの上限に達した場合、これらのリンクは他のモートがリンクを追加しないようにして帯域幅を確保します。より大きいネットワークやより深いネットワークでは、後述するように制限値に近づくことがより困難になります。

19.2 プロビジョニングの変更：IP VManager

VManager ネットワーク内の各 AP は、3 倍のプロビジョニングで 40 パケット/秒をサポートします。プロビジョニングを 1.5 倍に設定すると、最大スループットは 80 パケット/秒になりますが、すべての経路で 70%より高い安定性の確保が必要です。

マネージャ CLI を使用してプロビジョニング係数を変更するには、以下のように入力します（ここでは 1.5 倍）。

```
$> su becareful
#> config seti BWMULT=150
#> reset network [--reload]
```

19.3 プロビジョニングの変更：IP 組込みマネージャ

外付けの RAM がある IP 組込みマネージャは、256～284 スロット（平均 270）のランダム化されたスロットフレームに上りデータ専用のリンクが 223 あります（外付け RAM を使用しない場合は 150 リンク）。各スロットは 7.25 ミリ秒です。プロビジョニングが 3 倍の場合、これは 36.1 パケット/秒に相当します。プロビジョニングを 1.5 倍に設定すると、最大スループットは 72.2 パケット/秒になりますが、すべての経路で 70%より高い安定性の確保が必要です。

マネージャ CLI を使用してプロビジョニング係数を変更するには、以下のように入力します（ここでは 1.5 倍）。

```
> set config bwmult=150
> reset system
```

プロビジョニング係数をプログラム式に変更するには、**setNetworkConfig<bwMult>**というマネージャ API を使用します。

19.4 プロビジョニングの変更：WirelessHART

WirelessHART マネージャには、1024 スロットのスロットフレームに 737 の上りデータ専用リンクがあり、各スロットは 10 ミリ秒です。プロビジョニングが 3 倍の場合、これは 24.0 パケット/秒に相当します。プロビジョニングを 1.5 倍に設定すると、48.0 パケット/秒の最大スループットが得られます。

プロビジョニング係数を変更するには（ここでは 1.5 倍）、`/opt/dust-manager/conf/config/dcc.ini` にある `dcc.ini` ファイルの **link_oversubscribe** パラメータを以下のように指定します。

```
# Oversubscribe coefficient for link (1.0 no oversubscribing)
# Range: 1-100
 link_oversubscribe = 1.5
```

デフォルトでは、`dcc.ini` ファイルは存在しません。まだパラメータを変更していない場合は、`dcc.ini` ファイルを作成してプロビジョニング係数を変更する必要があります。必ず上記に示すディレクトリにファイルを作成してください。

20 アプリケーション・ノート：輻輳したネットワークのデバッグ

20.1 概要

SmartMesh ネットワークは、各モートが受け付けたすべてのパケットを各センサーから送出するよう設計されています。あるパケットがモートに受け付けられたがマネージャに到達しない場合、このパケットは**消失**したと分類され、ネットワークの**信頼性**にとってマイナスとなります。マネージャが報告する信頼性統計情報は、受け付けられたパケット数の合計に対する非消失パケット数の割合です。

可用性と呼ばれる別の重要な測定基準があります。可用性とは、センサーがモートにパケットを渡そうとしたとき実際に渡すことができた回数の割合のことです。SmartMesh ネットワークは、通常、100%の信頼性に加えて 100%の可用性を確保するよう十分なリンクが準備されますが、可用性はアプリケーション層の測定基準なので、可用性を直接測定しているわけではありません。モートがパケットを受け付けることができない唯一の機会は、モートのキューがパケットで満杯になっているときで、これは**輻輳**モートと呼んでいます。輻輳モートには、そのローカル・トラフィックと転送トラフィックをサポートする十分な上りリンクがありません。ネットワークが可用性を失う危険にさらされているかどうかを判断するには、輻輳モートを調べる必要があります。

20.2 サービスの順守

SmartMesh マネージャは**サービス**・モデルを使用してネットワーク内の帯域幅を割り当てます。このモデルでは、送信が必要なデータ量の算出を各センサー・アプリケーションが担当し、マネージャからの十分な帯域幅の要求を関連のモートが担当します。要約すると、アプリケーションの役割は以下の内容を実行することです。

1. 個々のデータ・フローについてパケット生成間隔を計算します。
2. これらすべてのデータ・フロー全体に対してサービスを要求します（これらは SmartMesh WirelessHART では個別に要求できますが、SmartMesh IP では 1 サービスだけです）。
3. このサービス、またはより高速のサービスがマネージャによって受け付けられたという確認を待ってから発行します。
4. 確認が到着しない場合は、タイムアウト後にサービスを再度要求します。

SmartMesh WirelessHART と SmartMesh IP では、サービス・モデルに関する詳細が異なる場合が多数あります。詳細については、それぞれのガイド（SmartMesh WirelessHART Services および SmartMesh IP Services）を参照してください。

20.3 可用性の推定

モートが周期的なデータを生成していることと周期が分っている場合は、マネージャで 15 分おきに受信されるパケットの数を推定できます。例えば、すべてのモートが 1 秒に 1 回報告している場合、15 分ごとに 900 データ・パケット

が予想されます。更に、モートは 15 分ごとに 3 つの健全性レポート・パケットを送信し、マネージャの要求および経路アラームに対する応答も送信する可能性があります。可用性は標準で 100%であるか、時間が経過すると大幅に低下するので、15 分間隔につき 903 パケット前後の数値がこの例では良好と考えられます。

間隔ごとの送信パケット数が少ないと、パケットがモートによってすべて受け付けられ、マネージャによって正常に受信されたかどうかを確認するのがより困難になります。報告回数の少ないネットワークはリンク数が少ないので、通常は待ち時間が長くなってパケットが次の 15 分間隔に押し込まれ、数パケットの増減が報告総数の大きな割合を占めることがあります。以下に示すモート統計情報の取得結果では、すべてのモートが 5 秒に 1 回報告しているので、1 回の間隔につき約 183 パケットが予想されます。1 回の間隔中に到着したパケットの数を知らせる PkArr 列のモート 11 の数値は、疑わしい低さです。以下に示す例では SmartMesh WirelessHART マネージャの統計情報を示していますが、SmartMesh IP ネットワークにも同じ考え方が適用されます。

```
> show stat short 0
It is now ................. 06/06/12 16:30:17.

This interval started at ... 06/06/12 16:15:00.

 --------------------------------NETWORK STATS-----------------------------------

PkArr  PkLost   PkTx(Fail/ Mic/ Seq) PkRx   Relia   Latency   Stability

 1806       0   5395(2430/   0/   0) 3098    100%    3.61s      54.96%
 ---------------------------------MOTE STATS-----------------------------------

 Id   PkArr   PkLst   PkGen   PkTer   PkFwd   PkDrp   PkDup   Late. Jn   Hop avQ  mxQ  me ne  Chg   T

  2     181       0     181       0     356       0      16   1.34   0    1   0    4   0  0    0  22

  3     182       0     182       0     194       0      20   1.49   0  1.2   0    3   0  0    0  22

  4     182       0     183       1      98       0      17   1.63   0    1   0    4   0  0    0  22

  5     182       0     182       0     329       0      18   1.46   0  1.4   0    4   0  0    0  22

  6     182       0     182       0      82       0      16   1.7    0  1.2   0    2   0  0    0  22

  7     154       0      --      --      --      --      42   20.7   0  2.6   -    -   -  -    -   -

  8     183       0     157       4      43       0      30   2.66   0  2.3   0    1   0  0    0  22

  9     183       0     182       0       0       0      30   3.39   0    2   0    2   0  0    0  22

 10     182       0     188       6      65       0      45   2.96   0  2.1   0    3   0  0    0  22

 11     166       0     169       1       0       0      42   17.2   0    3   1    4   0  0    0  22
```

ここでの信頼性は 100%です。PkLst 列の項目数は 0 ですが、アプリケーションはモート 7 および 11 からより多くのパケット数を予想していたので、これらのモートでは可用性が 100%より小さいです。一般に、センサーが送信しようとしているパケット数は分らないので、代わりに輻輳の兆候に注意する必要があります。

20.4 輻輳の識別

可用性の推定値に加えて、予想より長い待ち時間や、モートのキューの数値を統計情報で調べることで、輻輳を識別できます。おおよその目安として、キューの長さの平均値(avQ)が 0 より大きいモートは、どこかの時点で輻輳を観測している危険があります。キューの最大長(mxQ)が 4 以上のモートは、パケット生成間隔の間に深刻な輻輳に見舞われた可能性があります。最後に、輻輳モートの待ち時間は、通常は対等のモートの待ち時間より長くなります。上記統計情報でのモート 11 はこれらの基準をすべて満たしており、直前の 15 分間は間違いなく輻輳状態でした。

健全性レポートの欠落も輻輳を示している可能性があります。このシナリオでは、モートが健全性レポートを生成する時刻にモートのキューが満杯であり、モートは動作を完了することができませんでした。モートの統計情報では、これは上記のモート 7 の行のような形で現れます。キューの占有率に関するレポートを直接取得したわけではありませんが、それでも対等なモートより PkArr の値が低く、待ち時間が長いことが分ります。モートは健全性レポートの生成に失敗すると、カウンタを増やし続けて、次の健全性レポートには以前と同様に不足情報が要約されるように動作します。

モートは輻輳状態になると、その子モートがパケットをアンロードしようとするときに、子モートに NACK の送信を開始します。これにより、輻輳モートの子モートでの実効上り帯域幅が狭まるので、プロビジョニングの余裕が十分でない場合は、子モート自体が輻輳状態になる原因となる可能性があります。この進行はメッシュ全体にわたって繰り返されることがあります。このため、問題のモートより 3 ホップ深いモートは輻輳状態になる可能性があり、そのセンサーがオフに戻っていると認識します。この結果、マネージャがこのモートから受信するデータ・パケット数は予想より少なくなります。輻輳の発生源を見つけるため、各輻輳モートの上り経路を AP に向かってたどる必要があります。輻輳している最低ホップのモートが、その子孫モートの輻輳の推定発生源です。

20.5 帯域幅モデル

マネージャが各モートに与えようとするリンク数は、モートが安定性 100%のネットワークでローカル・トラフィックと転送トラフィックを理論上伝送する必要があるリンク数の 3 倍です。これは安定性が 33%に低下するまでネットワークが動作できることを意味していますが、安定性 33%とは、しばらくの間は 100%で、その後しばらくすると 0%になるということも多く、それは安定した「2 回失敗して 1 回成功」のパターンと比べると、正常なマルチホップ・データ収集に適していません。どこかに輻輳が存在する場合は、このモデルのどこかが壊れていて、輻輳の発生源については以下のいずれかが当てはまります。

- 経路アラームが原因で最近親を失った
- 経路安定性が 33%より低い
- 消費電力またはメモリの制約により、割り当て可能なリンクを使い果たした
- 子孫モートによる報告がそのサービス・レベル/基本帯域幅レベルで可能な数を超えている

経路アラームはマネージャ通知にサブスクライブすることでモニタできます。また、経路アラームは通常、極めて一時的な輻輳を引き起こしますが、モートに十分な数の良好な隣接モートがある場合、この輻輳は解消します。モートに十分な数の良好な隣接モートがない場合は、経路安定性が常に低い値となり、慢性的な輻輳が生じることがありますが、これは干渉が原因である可能性があります。指針については、「干渉の影響の特定および軽減」アプリケーション・ノートを参照してください。

マネージャ CLI で `show mote` コマンドを発行して、輻輳モートがそのリンク限界にどの程度近づいているかを確認することができます。特に、以下の 2 行を調べてください。

```
Neighbours: 27 (max 32). Links: 34 (max 100)
  Links per second: 4.882812 (limit: 11.648407)
```

ここで示されているリンクの数（34）は、最大値である 100 をはるかに下回っており、消費電力に起因する限界である 11.6 リンク/秒は、現在の 4.9 リンク/秒を優に超えています。いずれかのモートがリンク限界に近づいている場合は、これらのモートのすぐ近くに中継器モートを追加してください。

最後に、マネージャは各センサーがサービス・モデルを順守することを期待します。ネットワークの基本帯域幅で規定されている速度より高速にセンサーが報告する予定の場合、センサーはより高速のサービスを要求します。それでも、センサーがこれを適切に実行する保証はないので、引き続き潜在的な根本原因とみなされます。

モートがその要求サービス・レベルの限度を超えていないことを確認するため、`show mote` の結果をもう一度使用できます。

```
Bandwidth:
      output        planned  0.3906 current    0.3906
     global service          0.3614   delta   -0.0108
      local service goal     0.0250 current    0.0250
    guaranteed for services  0.0250 for child  0.3241 Free   0.0415
```

ここでは、ローカル・サービスの行を調べます。この最初の値 0.025 は、モートが要求したとマネージャが考えているパケット/秒の値で、サービスと基本帯域幅の両方を含みます。現在の値も同じなので、マネージャが実際にこの帯域幅を割り当てたことを示しています。差分の負の値が示しているのは、表面上はすべてのローカル・トラフィックおよび子孫モートのトラフィックをサポートするのに十分な帯域幅がこのモートにあるということです。0.025 パケット/秒では、15 分の間隔でモートが生成するデータ・パケットは、毎回最大でも 36 です。以前示したモートの統計情報（特に PkArr 列の情報）を使用して、モートによる生成数がこの最大レベルより多いことはなく、割り当てを超えないことを確認できます。

20.6 輻輳の緩和

SmartMesh ネットワークが数箇所で輻輳している場合は、ネットワーク全体の経路安定性が低すぎて正常に機能できない可能性があります。このシナリオでは、配置でのモートの密度を高くすることが望ましい対応です。これらの追加モートは、それまでに配置されているモートの間に広がり、潜在的な選択対象経路の数を増大できるので、その結果全体的な経路安定性が向上します。新しいモートが追加データを報告していない場合、必要な全放出帯域幅がこの解決策によって広がることはなく、むしろ既存のモートの平均消費電力が減少します。

配置の密度を高くすることができない場合は、輻輳モートでのリンクの数を増やすことができます。輻輳がグローバルである場合、ネットワーク内のすべてのモートでのリンクの数を増やす等価の方法が 2 つあります。プロビジョニングを増加するか、基本帯域幅を減少することができます。輻輳がネットワーク内の 1 つの分岐に集中している場合は、この分岐内にあるモートがより高速のサービスを要求している可能性があります。リンクの数を増やすと、リンク

数が増えたモートは、対応する消費電力が増えます。更に、これらが増加すると、アクセス・ポイントに余分な受信リンクが必要になります。割り当てに使用できる余分な帯域幅がアクセス・ポイントに存在しない場合は、ネットワークを 2 つに分割して追加リンクのサポートが必要になる可能性があります。

21 アプリケーション・ノート: 干渉の影響の特定および軽減

21.1 概要

SmartMesh ネットワークは、性能低下を最小限に抑えて干渉に耐えられるよう設計されています。干渉のある環境を干渉のない環境と比較した場合、平均のモート消費電力および待ち時間の増加は小さいと想定されます。ただし、一部のケースでは性能に大きく影響するのに十分なほど干渉が強いため、主に以下の症状を示すことがあります。

- モートがリセットする
- RSSI > -70dBm の場合でも、15 分の経路安定性の多数が 60%に満たない
- 深さが 3 ホップ未満のモートで、上りの待ち時間が 2 秒を超える
- 平均キュー占有率 > 1、または最大占有率 > 3
- モートのリセットまたは長い待ち時間により、信頼性が 99.9%を下回っている

干渉は、802.11g 無線ルータなどの帯域内干渉物、または WiMax などの極端に大きな帯域外干渉物の形を取ることがあります。スペクトラム・アナライザ(Wi-Spy などの割安な WiFi 検出器)を使用すると、混雑している周波数帯の領域を特定するのに役立てることができます。以下に例を挙げます。

- Bluetooth などの高速ホッピング干渉物が、一様に引き上げられたノイズ・フロアとして現れる。
- 802.11 WiFi ルータが複数のチャンネルにまたがる広範なピークとして現れる。
- 帯域外干渉物はまったく現れないが、それでもネットワーク内のレシーバーを飽和させる可能性がある。

スペクトラム・アナライザを結合した指向性アンテナが干渉の発生源を正確に特定するのに役立つことがありますが、スペクトラム・アナライザをまったく使用しないで干渉物の存在を推定することが可能な場合もよくあります。ネットワーク統計情報を使用して干渉物の存在を推測する方法も多く、また、干渉物が緩和に値するほどの重要な問題であるかどうかを判断することもできます。

21.2 RSSI と経路安定性の確認

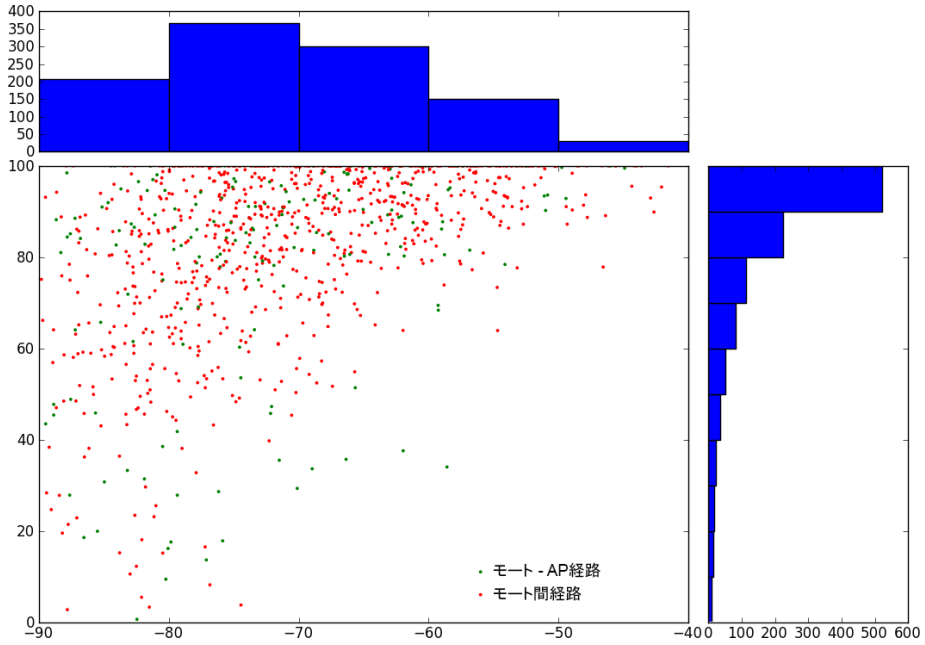
干渉に特に配慮する必要があるのは、ネットワークの経路安定性に影響する場合だけです。このケースに該当するかどうか調べるには、ネットワークからの健全性レポート・パケットを数時間取り込むことを推奨します。簡単な方法として、SmartMesh SDK の `HRListener.py` ツールを使用する方法があります。この情報を使用して滝型曲線をプロットすることができます。隣接モート健全性レポートに含まれる上り経路ごとに、滝の上に点を 1 つプロットします。このとき、報告された RSSI を X 軸にとり、15 分間の平均経路安定性を Y 軸にとります。例えば、2 つの親を持つモートがある場合、15 分ごとに隣接モート健全性レポートが 1 つ生成され、15 分ごとに 2 つの点がプロットされます。

以下に、ツールで生成された `receivedHRs.log` ファイルの抜粋を示します。

```
2014-07-09 11:52:08,233 [INFO] from 00-17-0d-00-00-3f-fc-04:
- Neighbors:
   - neighbors:
       -item 0
            - neighborFlag      : 0
            - neighborId        : 10
            - numRxPackets      : 45
            - numTxFailures     : 0
            - numTxPackets      : 0
            - rssi              : -58
       -item 1
            - neighborFlag      : 0
            - neighborId        : 1
            - numRxPackets      : 5
            - numTxFailures     : 545
            - numTxPackets      : 1275
            - rssi              : -72
```

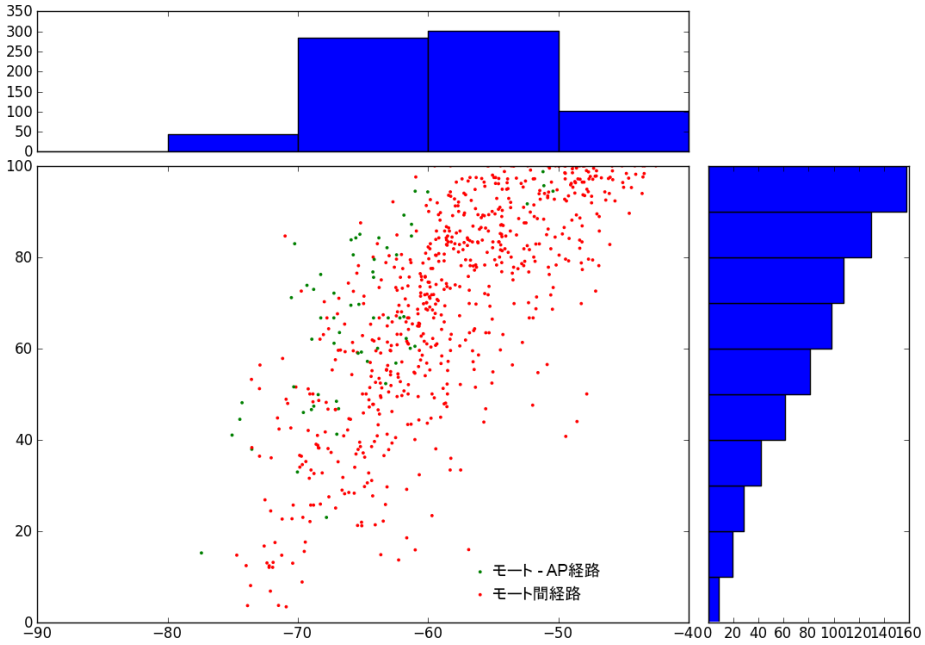
ここで確認するのは `numTxPackets` が 15 を超えるすべての経路なので、`neighborId` = 10 までの経路である「item 0」は対象外です。`neighborId` = 1 までの経路は対象となり、この経路の安定性は 1 - 545 / 1275 = 57% です。滝の上に点（-72dBm、57%）をプロットします。大規模ネットワークでは滝型曲線が密集する場合があるので、密度がわかりやすいように、縦と横のヒストグラムをプロットすることを推奨します。

良い滝型曲線は以下のような形をしています。



上側のヒストグラムでは、-90～-80dBm の範囲で使用された経路の数と、同様に-60～-50dBm の範囲で使用される経路の数を確認してください。右側のヒストグラムでは、経路の大部分が 80%以上であり、50%より低い経路は少ないことを確認してください。散布図では、-70dBm より良好なほとんどの経路は 60%以上の安定性であると予想します。モート間の経路とモート-AP 間の経路を分けることで、AP に固有のハードウェアまたは干渉の違いが見てとれます。

良い滝型曲線と次の曲線を対比してみます。



ここに示す曲線は全体が右側にシフトしており、-70dBm より低い経路がほとんど見当たりません。デバイスが微弱信号の受信に苦労しているので、このタイプの滝型曲線は不良レシーバーまたは干渉の典型例です。このネットワークが性能に対する顧客の期待を満足する可能性はまだありますが、このような配置ですべてのモートに十分な接続性を確保するよう特別な配慮が必要です。

開発中の開発者は、目標とする配置環境に似た「既知の良好な」環境で顧客固有のハードウェアを使用し、測定可能な干渉のない状況でテスト・ネットワークを配置する必要があります。このテスト配置で生成された滝型曲線は、どんなフィールド内データとも比較できる最高条件として使用します。特にアンテナの選択は受信感度に大きな影響を与えるので、すべての設計で比較の基準となるようなゴールド・スタンダードの滝型曲線を、Dust が 1 つだけ提供することはできません。

21.3 モートの待ち時間、キューの長さ、および信頼性の確認

一部の統計情報は、マネージャの CLI で `show stat` コマンドを使用してチェックできます。ここで該当するのは、ネットワークの信頼性と個々のモートの待ち時間の値です。以下の例では、信頼性は申し分なく、待ち時間の値（ミリ秒単位で表示）も 2 秒という前述の上限値より十分小さくなっています。

```
> show stat
Manager Statistics --------------------------------
   established connections : 1
   dropped connections     : 0
   transmit OK             : 1687
   transmit error          : 0
   transmit repeat         : 1
   receive OK              : 2028
   receive error           : 33
   acknowledge delay avrg  : 34 msec
   acknowledge delay max   : 112 msec
Network Statistics --------------------------------
   reliability: 100% (Arrived/Lost:   3677/0)
   stability:    72% (Transmit/Fails: 102701/28964)
   latency:    1300 msec
Motes Statistics ----------------------------------
   Mote   Received    Lost     Reliability  Latency   Hops
   #2          113       0     100%         240       1.4
   #3           95       0     100%         230       1.6
   #4           76       0     100%         170       1.3
   #5           51       0     100%         120       1.1
   #6           57       0     100%         160       1.1
   #7           97       0     100%         440       2.3
   #8           48       0     100%         140       1.0
```

デバイス健全性レポートも `HRListener.py` を使用して取得できますが、このレポートはモートのキュー・レベルと輻輳レベルに関する情報を提供します。以下に例を示します。

```
2014-07-09 11:52:42,542 [INFO] from 00-17-0d-00-00-3f-fd-57:
- Device:
   - badLinkFrameId     : 0
   - badLinkOffset      : 0
   - badLinkSlot        : 0
   - batteryVoltage     : 3160
   - charge             : 88
   - numMacDropped      : 0
   - numRxLost          : 0
   - numRxOk            : 2
   - numTxBad           : 0
   - numTxFail          : 0
   - numTxOk            : 35
   - queueOcc           : 33
   - temperature        : 25
```

はじめに、整数で表示される queueOcc の値について考えます。この整数の先頭 4 ビットは直前の 15 分間に経験した最大キュー長であり、末尾の 4 ビットは直前の 15 分間の平均キュー長です。ここで表示されている値の 33 = 00100001 であることから、最大キュー長は 2 で、平均キュー長は 1 であったことがわかります。このアプリケーション・ノートの最初に示したルールに従うと、この平均キュー占有値は、モートが輻輳していることを示しています。次に、numTxFail の値を確認します。これは、キューがいっぱいであったために、モートがローカルで生成されたパケットの受け取りを拒否した回数です。例ではこの値がゼロですが、ゼロではない場合、輻輳の兆候と考えられます。

21.4 軽減

干渉があることが特定されたら、干渉を軽減するための戦略がいくつかあります。適切な戦略は、干渉の時間的性質および強さとネットワークの性能目標によって異なります。ここからは、それぞれの戦略について詳しく説明します。ネットワークの動作を変更する戦略を有効にするには、ほとんどの場合、ネットワークのリセットが必要になります。

21.4.1 干渉の発生源の除去

このためには、指向性アンテナを使用して発生源を突き止めるか、他のデバイスがすぐ近くで送信している可能性とその内容を知ることが必要です。干渉物が法律で認められたもので、動かせないことも少なくありません。しかし、IT 部門が 2.4GHz 域外でのマルチバンド Wi-Fi の運用を受け入れてくれる可能性もあります。また、サードパーティの WiMax 干渉物による伝送が許可された上限を超えていたため、伝送の抑制を求めて法的措置をとることに成功した事例もあります。

21.4.2 ブラックリスト登録

干渉の影響する範囲がわずかである場合は、これをブラックリストに登録することができます。SmartMesh IP システムでは、ブラックリストを使用して、使用可能な 15 チャンネルのうち 7 チャンネルまで絞り込むことができます。詳細および注意点については、SmartMesh IP User's Guide の Channel Blacklisting（チャンネルのブラックリスト）セクションを参照してください。

21.4.3 中継器モートの追加

干渉による影響が最も深刻な経路を避けようとマネージャが試みても、モート側に代わりとなる受け入れ可能な親が存在しない場合があります。この場合、問題となっているモートと現在の親の間に中継器を追加することができます。信号が伝達される距離を半減することで、子モートの安定性を上げるのに十分な SNR の上昇が見込まれます。ネットワークに別のデバイスを追加しても、平均すると既存デバイスの経路安定性が高くなるので、平均電流は低下します。この戦略をとる場合、中継器の配置はマネージャから始めて外側に向けて進めることを推奨します。

21.4.4 親の増加

干渉によって安定性が低下しているネットワークでは、マネージャがすべてのモートとの間で信頼性の高い下り通信を維持することが難しいので、モートのリセットが発生する可能性があります。下り伝送は最終的にブロードキャスト・フラッディングにつながるので、ネットワーク内により多くの親を持つモートの方が、下りパケットを受信する可能性が高くなります。ネットワーク全体で親の数を変更するには、API コマンド **setNetworkConfig()**で `numParents` パラメータを指定します。親が増えるとモートの使用時間が増えるので、モート当たりの平均電流は増加します。LTC5800 の場合、親の数をデフォルトの 2 つから 3 つに増やしても使用中のルータへの影響は最小限にとどまり、平均電流の増加は約 15%です。データ量の少ないリーフ・モートでは、平均電流が最大 32%上昇する場合があります。干渉が極めて大きい場合、親の数を 4 つに増やすことも可能です。この場合、同等の平均電流の上昇が追加で見込まれます。

21.4.5 下りの再試行回数とタイムアウトの増加

消失モートの正確な特定が優先事項ではない場合、消費電力を大幅に増加させることなく、下りの信頼性を高める方法がもう 1 つあります。マネージャがモートの消失を宣言する前に再試行する最大回数を増やすことです。マネージャはデフォルトで、各パケット送信を最大 5 回試行します。マネージャの CLI で次のコマンドを実行すると、この上限を 2 倍に増やすことができます。

```
su becareful
seti ini numretr 10
```

この変更だけで、だいたい 5 ホップ以下の深くないネットワークに回復力を持たせることができます。影響を受けているネットワークがこれよりも深い場合、下りパケットが宛先に到達する前にタイムアウトする可能性があります。マネージャの再試行タイムアウトを長くすると、その後でモートへの送信を試行する間隔が長くなるので、この場合もモートの消失を特定するまでの時間が延びる効果があります。これらのシナリオでは、深い IP ネットワークの構築アプリケーション・ノートに従ってネットワークを構築することを推奨します。

これらの変更によってリセット・レートがどれだけ低下するかを正確に示すことは困難です。リセット動作は、干渉が最大になった時点でのネットワーク内の最も低品質な経路という、2 つの予測しがたい限界によって決まるからです。ただし、干渉の大きい特定の配置で、これらの変更によって 1 日のリセットが約 10 回から 1 回未満に減少したケースもあります。

21.4.6 プロビジョニング係数の増加

干渉によって待ち時間が長くなっているが、モートのリセットは発生していない場合、モートの上りリンクを増やすと効果があります。上りリンクが増えると輻輳が減り、主にマルチホップ・ネットワークで低い経路安定性を克服するために役立ちます。プロビジョニングのデフォルト値の 300%は、モートが、送信する必要のある各パケットの送信を平均で 3 回以上試行するという意味です。API コマンド **setNetworkConfig()**で `bwMult` パラメータを指定し、この値を400%というより慎重な値に増加することができます。

LTC5800 デバイスでこの値を 300%から 400%に増やす場合、リーフ・モートの平均電流が大幅に増加することはありませんが、ルータの平均電流は約 5%増加する場合があります。

極端なケースではプロビジョニングを最大 600%に増加できますが、この場合ルータの電流は約 15%増加します。

21.4.7 新経路に対する RSSI 閾値の増加

新しく検出された経路の RSSI が-80dBm を上回る場合、マネージャはデフォルトで、この経路をテストするためにリンクを追加しようとします。滝型データにより、-70dBm を下回る経路では良好な安定性がほとんど得られないことが示される場合、INI の設定でこの新しい経路の閾値を適宜増加できます。

```
su becareful
seti ini rssith -70
```

この場合のトレードオフとして、値を慎重に設定しすぎると、必要以上にホップ数の多いネットワークが構築され、平均の電流および待ち時間が増加します。

21.4.8　狭帯域フィルタの追加

2.4～2.48GHz 帯以外で干渉が発生している場合、狭帯域（例：BAW）フィルタを追加すると受信する干渉を減らすことができます。このソリューションでは、以下のガイドラインに従う必要があります。

- シンボル間干渉への影響を最小限に抑えるため、群遅延がほぼ一定のフィルタを選択します。
- 通常、フィルタを追加すると 2dB の挿入損失が発生し、これが受信経路と送信経路の両方に影響するので、経路全体では 4dB のリンク・バジェットを損失することになります。
- フィルタ・メーカーの指定に従い、適切な接地を行います。
- デバイスの全動作範囲（温度、電圧など）に対応する適切な通過帯域を選択します。ここでの目標は、上位チャンネルおよび下位チャンネルの性能を低下させることなく、予想される干渉をフィルタリングすることです。

22 アプリケーション・ノート：正確なタイムスタンプの取得

22.1 時間

Dust ネットワークのすべてのデバイスは時間の感覚を共有しています。これにより、センサー・アプリケーションは、ネットワーク時刻を使用してセンサー測定の正確なタイムスタンプ処理を実現できます。理想条件下では精度を数十 µs 程度に厳しくすることができます。処理は製品系列全体で同様です。

1. センサー・プロセッサは測定を実施し、モートの時刻ピンにストローブ信号を送出します。
2. モートはその現地時刻を（UTC および ASN で）返します。
3. センサー・プロセッサはタイムスタンプおよびデータをパケットに書き込みます。
4. モートはパケットをマネージャに転送して、ホスト・アプリケーションに送出します。
5. ホストは必要に応じてデータ通知を処理し、ネットワーク時刻と「絶対」時刻の差に対応します。

測定のタイムスタンプ精度はシステム内での種々の不確実性によって決まりますが、その内容についてはこのアプリケーション・ノートで説明します。

22.2 参考資料

- RFC 5905 は、Network Time Protocol（NTP）のバージョン 4 を規定しています。
- IEEE1588-2008 は、Precision Time Protocol（PTP）のバージョン 2 を規定しています。
- IEEE C37.238 は、パワーシステムで IEEE1588 を使用するための規格です。この規格は、広範囲にわたって隔てられた設備を同期化するためにイーサネットの使用や様々な設定について規定しています。

22.3 VManager の IP システム

VManager ネットワークで正確なタイムスタンプが必要な場合は、アクセス・ポイントを GPS クロックに接続することを推奨します。こうすることで、ASN は UTC 時刻に 10µs 未満で追随するので、モートを含むネットワーク全体でこの分解能を使用できます。このシステムを使用した場合、このアプリケーション・ノートに記載した時間差を考慮に入れる必要はありません。

22.4 組込みマネージャの IP システム

ネットワーク時刻を使用する手順のフローチャートを図 1 に示します。SmartMesh IP マネージャは起動時に、「UTC 時刻」を 2002 年 7 月 2 日 20:00:00 から開始します（マネージャの API コマンド **setTime** は廃止済みのため、ネットワーク形成前に時刻を設定することはできません）。そのときからマネージャは「絶対」時刻から徐々に離れていき、現時点では時刻を継続的に修正する仕組み（例えば、SmartMesh WirelessHART マネージャで利用できるような NTP）が存在しないので、**マネージャの UTC 不確実性**が生じる原因になります。マネージャの時刻ピンを使用するこ

とにより、ホスト・アプリケーションはこのずれを修正できます。時刻は ASN と UTC の両方でモートに返されるので、正確なタイムスタンプを得るためにマネージャに有効な UTC 時刻を設定する必要はありません。

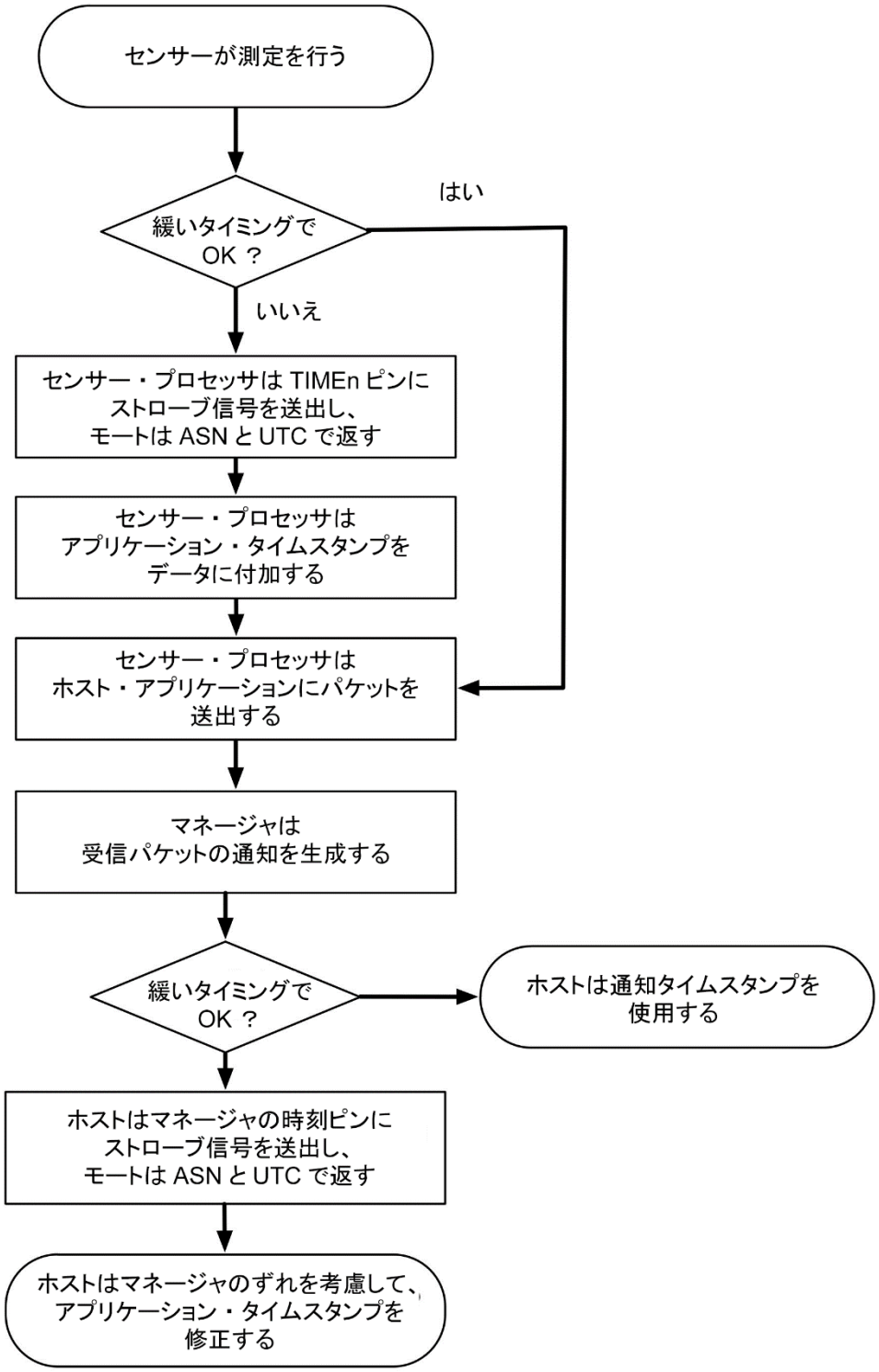


図 1 – Eterna LTC5800/5901/5902 マネージャ・ネットワークでのタイムスタンプ処理

22.5 緩いタイミング

緩い（数百ミリ秒）タイミングだけが必要な場合は、アプリケーションのタイムスタンプは必要ありません。ホスト・アプリケーションは **data** 通知または **ipData** 通知からのタイムスタンプを使用できます。このタイムスタンプには**キュー操作の不確実性**が存在し、それは数十ミリ秒になることがあります。このタイムスタンプにもやはりマネージャ UTC の不確実性が存在しますが、これについてはタイムスタンプ処理されたデータに照らした修正を後で説明します。

22.6 厳しいタイミング

1 ミリ秒未満のタイミングが必要な場合、センサー・プロセッサは測定を実施し、モートの時刻ピンにストローブ信号を送出してネットワークのタイムスタンプを取得し、パケットに組み込むことができます。モートはその現地時刻を（UTC および ASN で）返しますが、これには 2 つの不確実性があります。1 つは**モートの現地時刻の不確実性**で、これは経路安定性、モート間の温度のばらつき、トポロジなどの関数です。もう 1 つはモートの**時刻ピンの不確実性**で、HW ファミリのデータシート特性です。パケットには ASN または UTC のタイムスタンプを組み込むことができます。ASN を使用すると、UTC 時刻が実際に正確ではないことがより明確になります。

アプリケーション・プロセッサは、データ通知を受信したらマネージャの時刻ピンを起動する必要があるので、現在の ASN 時刻または UTC 時刻をマネージャの**時刻ピンの不確実性**の範囲内に入れます。これにより、アプリケーションはパケット内のタイムスタンプに加えてマネージャからの現在のタイムスタンプを使用することで、**マネージャの UTC 不確実性**を取り除き、パケットの絶対タイムスタンプがホストの時刻の精度範囲内に入るよう計算することができます。

22.7 最も高精度なタイミング

最高のタイミング精度を得るには、不確実性の原因をすべて考慮し、可能な場合は最小限に抑える必要があります。

- 通過時間（待ち時間）があると**通過の不確実性**が加わります。これはネットワーク・トポロジとネットワーク活動の関数です。活動が非常に少ないネットワークでパケットが生成された場合、そのパケットがマネージャに到着するのに 30 秒かかることがあります。マネージャのずれが 50ppm である場合は、通過時のマネージャのずれにより、1.5 ミリ秒の不確実性が計算値に加わります。アプリケーション・プロセッサがマネージャの時刻ピンに定期的にストローブ信号を送出してマネージャのずれを測定する場合は、通過の不確実性を除去できます。その後、アプリケーション・プロセッサは現行の UTC オフセットを計算して、マネージャの UTC 不確実性の他に、通過時間に起因する不確実性を除去します。ただし、標準的な条件下では、通過の不確実性は数マイクロ秒です。
- サンプルの取得およびモートの時刻ピンへのストローブ信号出力に関連した、センサー・プロセッサ内での遅延または不確実性については、判断して説明責任を負うインテグレータ次第です。
- 急な温度勾配は、**モートの現地時刻の不確実性**に影響することがあります。この誤差は、温度勾配のあるモートのすべての子孫モートに伝播します。この影響によってシステムの要求を超える誤差が発生する可能性がある事例では、アプリケーションがすべてのモートの温度を測定して、大幅な温度上昇が検出された場合にアラームを報告することができます。その後、ホスト・アプリケーションは、これらのアラーム期間中のタイムスタンプがあるデータを無視するか、すべてのモートが一定の温度になっている時間帯に取得した測定

値より大きな不確実性をアラーム期間中のデータに割り当てることを選択できます。安定した温度で動作するフラット・トポロジ・ネットワークは、モートの現地時刻の不確実性が最低になります。

22.8 IP 不確実性の定量化

以下に示すのは、IP ネットワークでの不確実性の値です。

- **時刻ピンの不確実性**– Eterna ベースのマネージャまたはモートでは、最も厳しい条件で+/-1 µs です。
- **モートの現地時刻の不確実性** – **h** ホップにあるモートの場合、標準的な不確実性は $0.6h^{1/2}$ マイクロ秒になり、最も厳しい条件で $30h$ マイクロ秒になると予想されます。更に、規格限界値である最大 8°C/分の温度勾配は、1 回の配信で 1.5µs·min/°C 増加します。
- **マネージャの UTC 不確実性** – 水晶振動子の特性を評価済みの場合、稼働時間は最も厳しい場合で+/-50µs/s、標準で+/-2µs/s です。
- **通過の不確実性** – 水晶振動子の特性を評価済みの場合、パケットの通過時間は最も厳しい場合で+/-50µs/s、標準で+/-2µs/s です。
- **キュー操作の不確実性** – パケットがモートによってアクノリッジを返されると仮定した場合、最も厳しい場合で 50 ミリ秒未満です。関係があるのは、ネットワーク層のタイムスタンプを使用する場合だけです。

22.9 WirelessHART（Linux SBC ベース）システム

Linux SBC ベース（PM2511/DN2511/LTC5903）のマネージャは、マネージャを絶対時刻基準に同期させ続けるため NTP をサポートしています。マネージャは UTC（グローバル）時刻を取得するため、NTP サーバに接続されています。グローバル時刻に関するマネージャの概念の精度は、**マネージャの UTC 不確実性**によってその特性が決まります。NTP サーバを専用接続回線に接続すると、IP ネットワークが不確実性に与える影響を取り除くことができます。

ネットワーク時刻（ASN）はアクセス・ポイント（AP）とは独立して維持されます。マネージャは AP の時刻ピンにストローブ信号を送出して AP の時刻を測定し、得られた時刻をマネージャ独自の推定値と比較します。この測定値には AP の**時刻ピンの不確実性**が含まれており、それはどちらの AP（DN2510 または Eterna）が使用されているかの関数です。AP の時刻が 100 ミリ秒（**max_utc_drift** という ini パラメータで設定可能）を超えて異なっている場合、マネージャは新しい UTC 時刻のマッピングを低信頼性のブロードキャストを介してすべてのモートに対して強要します。**max_utc_drift** の値を小さくすると、下りのメッセージが増加します。例えば、50ppm のずれがある LTC5800 ベースの AP の時刻をマネージャの UTC 時刻から 100 ミリ秒以内にするには、約 30 分に 1 回の DS メッセージが必要です。これを 10 ミリ秒に維持するには、3 分に 1 回のメッセージが必要です。

センサー・プロセッサがモートの UTC タイムスタンプを使用している場合は、UTC 不確実性に **UTC マッピングの不確実性**を追加する必要があります。モートが数回の UTC 更新に失敗し、そのため AP とのずれが 200 ミリ秒を超える可能性があります。理論上は ASN タイムスタンプを使用すると UTC マッピングの不確実性を排除できますが、WirelessHART マネージャは正確な ASN 時刻を取得する手段を備えていません。**getTime** マネージャ API は、マネージャが測定した最後の ASN/UTC マッピングを返すので、最大で **max_utc_drift** ミリ秒の誤差が発生する可能性があります。

不確実性の発生源の多くは、現在の SmartMesh WirelessHART マネージャでは制御することも原因を特定することもできないので、使用できるのは緩いタイミング（数百ミリ秒）だけです。

22.10 WirelessHART 不確実性の定量化

以下に示すのは、WirelessHART ネットワークでの不確実性の値です。

- **マネージャの UTC 不確実性** – これはマネージャ、NTP サーバ、および IP ネットワークの待ち時間の関数です。AP の関数ではありません。この測定値は低トラフィックのイーサネット・ネットワークでは約 1 ミリ秒になりますが、イーサネット・ネットワークが混雑状態になると 100 ミリ秒近くまで上昇することがあります。
- **時刻ピンの不確実性** – これは DN2510 ベースの AP またはモートでは-60/+600µs になり、Eterna ベースのモートでは+/-1 マイクロ秒になると予想されます。
- **UTC マッピングの不確実性** - モートが数回の UTC 更新に失敗し、そのため AP とのずれが最大+/-300 ミリ秒になる可能性があります。
- **モートの現地時刻の不確実性** – DN2510 ベースの AP を使用した場合、**h** ホップにあるモートでは、平均で $45\mathbf{h}^{1/2}$ マイクロ秒近辺になり、最も厳しい条件で $150\mathbf{h}$ マイクロ秒になると予想されます。LTC5800 ベースの AP を使用した場合、平均で $1.2\mathbf{h}^{1/2}$ マイクロ秒になり、最も厳しい条件で $30\mathbf{h}$ マイクロ秒になると予想されます。更に、規格限界値である最大 8°C/分の温度勾配は、1 回の配信で 3µs·min/°C 増加します。

22.11 同期イベント

同期測定または同期イベントもすべての製品で可能ですが、手順が若干異なります。

- アプリケーションはマネージャの **getTime** API 呼び出しを使用して現在の ASN を取得します。
- ホスト・アプリケーションは、(将来の)イベント時刻が書き込まれているアプリケーション層のパケットをブロードキャスト*します。メッセージは配信を確実に行うため、3 回送信されます。**すべてのモートは将来の ASN を使用してイベントを同期化できます。すべてのモートが将来の ASN を受信し、イベントに備える時間を確保できるように、将来の ASN として十分に大きな値を選択します。120 秒より大きな値が安全です。
- モートはパケットを受信し、アプリケーション層パケットが入っている通知をセンサー・プロセッサに送信します。各センサー・プロセッサは、モートの時刻ピンを使用して現在時刻を確認します。
- 各センサー・プロセッサは、イベント、測定、作動の時刻がいつかを適宜確認します。センサー・プロセッサが非補償型の 32kHz 水晶振動子を時刻の基準(+/- 150 PPM)として動作させていると仮定すると、120 秒タイマでの絶対サンプル時刻のデバイス間のばらつきが+/-15 ミリ秒まで広がる可能性があります。このためには 24 ビットのタイマが必要であることに注意してください。このレベルの同期が許容できない場合、センサー・プロセッサはモートを定期的にポーリングして現在の ASN を取得し、それに応じてタイマを調整できます。ASN の差分および時間の変換は次式のように行われます。

差分時間 = ASN の差分 * スロット長

ここで、SmartMesh WirelessHART のスロット長は 10 ミリ秒であり、SmartMesh IP のスロット長は 7.25 ミリ秒です。

- 各センサー・プロセッサは、イベントによって生成されたすべてのデータに対して上記のタイムスタンプ・ロジックをオプションで実行できます。

*宛先(IP メッシュ層または WirelessHART ネットワーク層)アドレスは 0xFFFF です。**コマンドを受け取るすべてのモートの待ち時間と可能性のトレードオフを考慮すると、低信頼性の下り伝送に対しては 3 回の再試行が選択されます。再試行の回数が増えるとこの可能性は高くなりますが、待ち時間が長くなります(つまり、将来の ASN はより広げる必要があります)。アプリケーション・レベルのアクノリッジがセンサー・プロセッサによって返される場合は、選択的な再試行(による重複の削減)が可能です。センサー・プロセッサは(将来の ASN として同じものが記載された)重複コマンドを破棄する必要があります。

23 アプリケーション・ノート：複数のマネージャを使用した大規模ネットワークの構築

23.1 大規模な配置

1 つのマネージャがサポートできる数より検出点の数が多い実装形態があります。現在では、複数のマネージャを置いて実装範囲をカバーしています。これらのマネージャとそのモートの一部またはすべてに、重複した無線到達範囲があることが予想されます。可能な場合は、「共同設置」マネージャを互いに隣接して配置するのではなく、数メートル離して配置する方が実践的に優れています。これはマネージャのアクセス・ポイント間の無線干渉を避けるためです。近辺にあるトラフィックの合計が安全なレベルに維持されていれば、複数の共同設置マネージャが配置された設備は完璧に機能します。

各マネージャはマネージャ固有のネットワークを制御します。また、モートを種々のネットワークに分ける非常に重要な方法が 2 つあります。第 1 に、各ネットワークにはネットワーク ID があります。この値はネットワークが送信したすべての通知内にあり、モートが参加したらパケットにフィルタを適用するために使用します。第 2 に、マネージャは、ネットワークへの参加が可能な各モートの MAC アドレスと参加鍵を指定するアクセス制御リスト（ACL）を保持できます。ネットワーク ID と ACL を組み合わせて様々な方法で使用することにより、大規模な配置でのモートを小規模なネットワークに分割することができます。ネットワーク ID を制御した場合は、参加先のネットワークを決定する負担がモートにかかります。ACL を制御した場合はマネージャが決定できますが、この方法は、ACL に登録されていないネットワークに参加しようとしているモートにとっては不都合な可能性があります。

使用するソフトウェア・ライセンスによって、SmartMesh WirelessHART マネージャは最大 250 モートまたは 500 モートをサポート可能であり、SmartMesh IP 組込みマネージャは最大 32 モートまたは 100 モート（外付け RAM 使用時）をサポートできます。これより多くのモートが必要な配置では複数のマネージャが必要になるのは明らかですが、配置全体の放出帯域幅を広げる目的でも複数のマネージャを配置できます。各 SmartMesh マネージャは 20～36 パケット/秒の放出帯域幅をサポートしているので、例えば、100 モートが 1 秒に 1 回データを報告する配置は、5 つの SmartMesh マネージャで安全に実行できます。

⚠ VManager ベースの SmartMesh IP ネットワークは数千単位のモートをサポートできます。大規模な配置では VManager の使用を推奨しますが、このアプリケーション・ノートでは、WirelessHART ネットワークが必要な場合や、SmartMesh 組込みマネージャが既に使用されている場合のソリューションについて説明します。

23.2 RF の制限事項

SmartMesh WirelessHART の場合は、存在するチャンネル数が 15 で 1 秒当たりのスロット数が 100 です。これは、1500 パケット/秒の絶対最大数を様々なチャンネル上であるいは様々な回数送信するには十分な**セル空間**です。複数のマネージャが同じ無線空間を共有している場合、マネージャはスケジュールも正確な時間の感覚も互いに共有しないので、トラフィックが重複した場合に衝突が発生します。マネージャを同じ場所に設置している場合はセル空間の約 25%を使用するのが安全であることが経験的に分っているので、この場合は、全放出帯域幅が 375 パケット/秒のポイントまでマネージャを追加しても安全です。一例として、4 つの SmartMesh WirelessHART マネージャを使用して 1800 モートが配置されており、各モートは 30 秒ごとに温度を報告するよう設定されているとします。この場合の全放出帯域幅の要件は 1800/30 = 60 パケット/秒なので、この配置は RF 制限値を十分に下回っています。

SmartMesh IP では、チャンネル数は同じ 15 ですが、1 秒当たりのスロット数は 138 です。同じ計算を実行すると、共同設置マネージャの安全な制限値は 517 パケット/秒です。

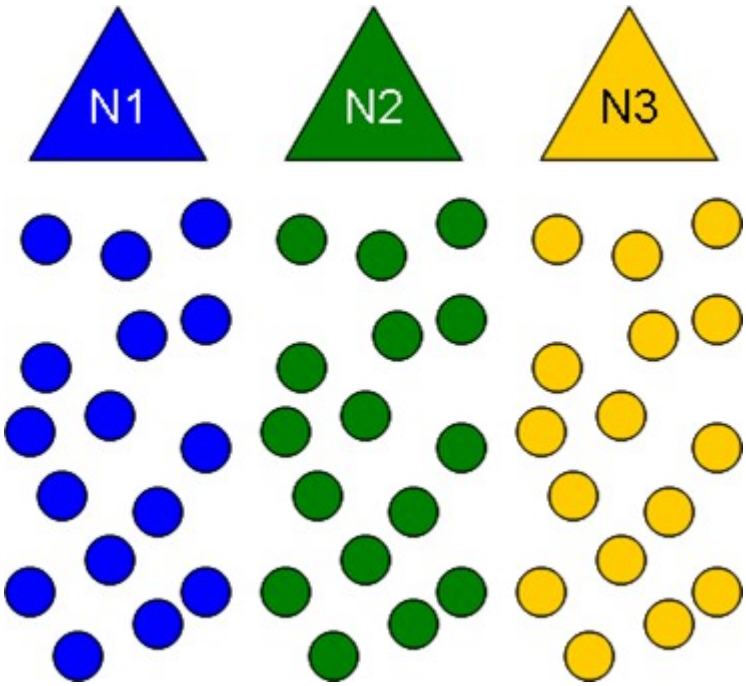
重複ネットワークのトラフィックが増加するにつれて、ネットワーク間での衝突が経路安定性のわずかな減少として現れます。SmartMesh マネージャには、ある 1 つの経路を完全に遮断するのではなく、すべての経路に均等に影響する傾向が強くなるように衝突を分散させる軽減策があります。Dust ネットワークはデューティ・サイクルが低く低消費電力なので、他の Dust ネットワークからの干渉は、WiFi のような他の形態の干渉よりも被害が少なくて済みます。

23.3 理想的な配置のガイドライン

理想的な事例では、使用可能なマネージャの間でモートが均等に分けられます。例えば、48 のモートと 3 つの SmartMesh マネージャがある場合、各マネージャには 16 のモートが与えられます。これを実行するには、いくつかの段階があります。

- 3 つの固有ネットワーク ID が選択され、各マネージャに 1 つずつ与えられる
- マネージャごとに 16 のモートが指定され、該当するネットワーク ID でプログラムされる
- 各マネージャは 16 のモートの MAC アドレスおよび参加鍵が登録された ACL でプログラムされる
- 対象の領域では、16 モートの各ネットワークが配置ガイドラインに従うようにモートが配置される

下の図では、ネットワーク ID で分けられた 3 つのネットワークを異なる色で表しています。モートの物理的な配置は、その他のネットワーク内のモートとは関係なく、色分けされた各ネットワークの接続状態が十分になるように行われます。



モートは通知をネットワーク ID 別に選別するので、参加処理中のモートが参加要求を送信しようとする相手は、参加処理中のモートとネットワーク ID が同じ親だけです。モートには正しいネットワーク ID が設定されているが、マネージャの ACL にそのモートが登録されていない場合（このシナリオでの誤り）、モートは参加しようとしますが参加要求パケットはマネージャによって廃棄されます。この場合、モートはその再試行と再リセットを実行し尽すまでに数分かかります。

この方法は最も安全でネットワーク形成が促進されますが、最適な性能にするには設定時間を費やすことが必要です。各モートには正しいネットワーク ID を設定する必要があり、各モートは現場の正しい区域に取り付ける必要があります。各マネージャは正しい ACL を取り込む必要があります。いずれかのネットワークで代替デバイスまたは中継器が必要な場合は、代替モートに正しいネットワーク ID を設定し、正しいマネージャに ACL 項目を与えてから新しいデバイスが参加できるようにする必要があります。更に、モートが配置空間内のある場所から別の場所に移動する場合、モートはその現在のネットワーク ID をそのまま残すことがあるので、新しい場所で別のネットワーク ID を使用して再参加できない場合があります。

23.4 モートの動作 - 探索の使用

モート・アプリケーションは、参加の前に API コマンド **search** を使用して、近辺にある様々なネットワーク ID をスキャンできます。このモードでは、モートはネットワーク ID に関係なく、**すべての**通知を受信します。モートは、受信する通知ごとに送信元のネットワーク ID、信号強度、および送信元の深さ（ホップ数）を報告します。これにより、モートが参加を要望するネットワークを自動制御で選択するための十分なデータがアプリケーションに与えられます。その後、アプリケーションはモートのネットワーク ID を設定して、**join** コマンドを実行します。この場合、モートは特定のネットワーク ID に対する通知のリスニングを開始します。この結果、モートはこの規定のネットワーク ID だけを参加の対象にしようとします。最初の起動時にモートが試行するべき望ましいネットワーク ID を使用して、アプリケーションをプログラミングすることを推奨します。タイムアウト後にこのネットワーク ID からの通知が受信されない場合、アプリケーションはモートを**探索**モードに設定して、代替ネットワークに参加するための情報を収集します。また、この

処理では一定の可動性が許容されます。ネットワーク ID が異なる隣接モートを使用して移動およびリセットを行うモートは、この新しいネットワーク ID を検出し、複数マネージャの配置内で別のネットワークに再参加できます。

23.5 マネージャの動作 - 様々なネットワーク ID と共有 ACL

⚠ ACL 項目が何千もある SmartMesh IP 組込みマネージャを使用するためには、外部 SRAM のインストールが必要であり、またマネージャのバージョンが 1.4.1 以降である必要があります。

SmartMesh マネージャは、数千のモート MAC ID および参加鍵を ACL に格納できます。複数マネージャの配置を計画している場合、例えば 1000 モートで 4 つのマネージャの場合は、各マネージャに最大 1000 モート分の ACL および参加鍵、またマネージャ固有の独立したネットワーク ID が与えられます。その結果、前のセクションで説明したように、モート・アプリケーションは **search** API を使用して、この領域内のすべてのネットワークの通知をスキャンできます。

または、複数のマネージャの配置を実行するのに、同じネットワーク ID で異なる ACL を使用してもかまいません。この場合、各マネージャは、受け付けられるモートがどれかを認識している必要があります。この方法を使用するのが妥当なのは、モートが予め定義されたマネージャに参加することが望ましい場合だけです。例えば、300 のモートが 6 つの SmartMesh WirelessHART マネージャの間で均等に分れている場合、各マネージャには異なる 50 モートの ACL が与えられ、最終的にどのモートがどのマネージャに制御されるかはこの ACL によって決定されることがあります。この方法を単独で使用するデメリットは、間違ったマネージャに参加しようとしているモートには、そのモートが拒否されていることが分らないということです。このモートは、それが通知を受信した親を通じて参加要求パケットを送信します。また、このモートはその親から適切なリンク層 ACK を受け取ります。参加側のモートは、マネージャからの応答を待っている間はネットワークとの同期を維持しますが、参加側のモートが ACL に登録されていないので、マネージャは参加要求を廃棄します。モートがリセットして再度参加を試みるまでには、長いタイムアウト時間がかかります。しかし、モートがいったん工場から出荷されると、モートに対してネットワーク ID を設定するのは簡単ではない場合があります。また、すべてのネットワークは同じネットワーク ID を使用して稼働する必要があるので、均衡のとれたネットワークを実現するには ACL を区切ることが唯一の方法です。

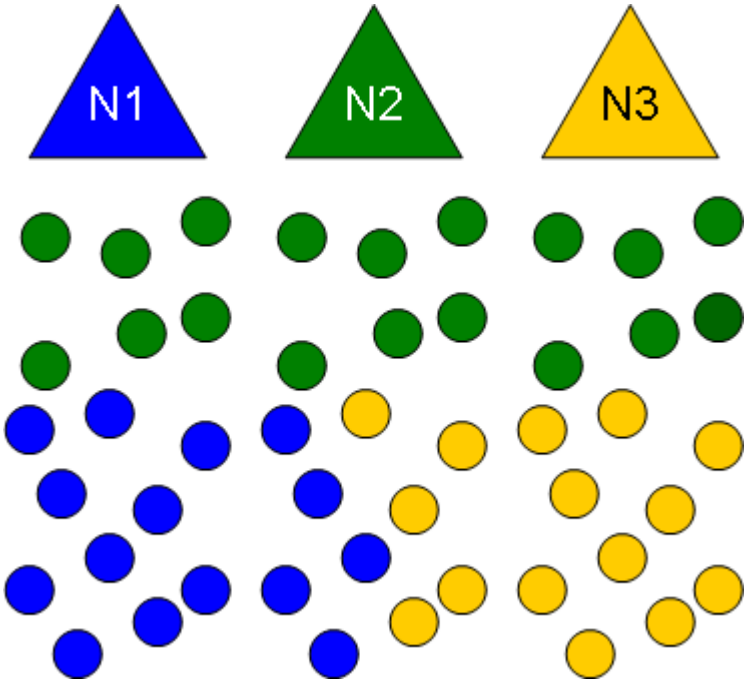
各ネットワークが範囲と接続性について配置のガイドラインに従うことを前提にすると、ACL を区切るだけでは、最終的にはすべてのモートが規定のネットワーク内に帰着するので、少し時間がかかる場合があります。モートが元のネットワークの外側にある配置の新しい区域に移動する場合、新しい区域のマネージャが新しい ACL 項目でプログラムされない限り、そのモートは再参加できません。少数のモートの可動性を有効にするため、配置内のすべてのマネージャは、これらの可動モートを ACL に登録できます。その後、ネットワークから離れる可動デバイスは、新しい通知を受信後にリセットして新しいネットワークに再参加します。

23.6 複数マネージャの配置に関する危険

採用する手法を決定するときに評価する必要がある危険性がいくつかあります。

すべてのモートおよびマネージャに単一のネットワーク ID を設定し、ACL をまったく区切らないことも可能です。単一のネットワーク ID を使用した場合、モートは通知の受信対象となったネットワークがどれであっても積極的に参加します。最初に通知を開始したマネージャが参加の雪崩効果を引き起こす傾向があるので、モートが利用可能なマネージャに対して均等に割り当てられない可能性があります。複数のマネージャ群に近いすべてのモートが参加することで、一般的なマネージャがモートの最大数または帯域幅の最大値に到達し、マネージャ群から離れているモートが立ち往生する可能性があります。この方法を採用する場合は、十分に使用されていないマネージャにモートが参加することを期待して、マネージャが限度に達しているのでモートを自動的にリセットすることをアプリケーションが認識できる必要があります。

配置を工場にまかせる前に、個別のネットワーク ID および ACL を適切に使用して個々のネットワークを適切に区分し、ネットワーク 1、ネットワーク 2 など別々の枠に入れることができます。ただし、モートを適切に設置するために配置時にもやはり注意が必要です。モートを個別に配置すると、マネージャの近くのネットワーク 1 の枠からすべてのモートが選ばれる場合があり、そのネットワークのマネージャの範囲を超えたネットワーク 2 のモートは行き場を失います。ネットワークの分散について何らかの試みを行う場合でも、全体としての配置だけにとどまらず、各ネットワークについて個別に接続性配置ガイドラインに従うよう注意する必要があります。ボトルネックには特別に注意を払う必要があります。ネットワーク 1 からのモートの周囲に多くのモートが存在する場合がありますが、それらがすべて他のネットワークからのモートである可能性があります。この場合は、中継器を設置するために追跡調査の訪問がときどき必要です。また、これらの中継器は特定のネットワーク ID で適切にプログラムする必要があります。



ネットワークの区切りが不適当だと、上の図に示すようにボトルネックや枯渇につながる恐れがあります。ここでは、緑のマネージャがその近くにあるモートをすべて参加させており、黄色のマネージャと青いマネージャは、それぞれの対象モートがあまり近くにないので参加させることができません。

23.7 ネットワーク ID と参加鍵の無線による設定

アプリケーションは、マネージャから無線でネットワーク ID と参加鍵を設定できます。一部の顧客は工場でのテスト・ビルド時に固定のネットワーク ID および参加鍵を使用し、その後、配置に備えてネットワークをパッケージ化する前に設定を固有の値に再プログラムします。これが特に役立つのは、再プログラミングのために物理的に接続できない密封されたモート・パッケージを持つ顧客です。新しい値はモート／マネージャのリセット後に有効になります。

24 アプリケーション・ノート：SmartMesh Power and Performance Estimator の使用

24.1 概要

このアプリケーション・ノートでは、SmartMesh Power and Performance Estimator を使用して SmartMesh ネットワークがどのように動作するかについておおまかな結論を出すためのいくつかの方法について説明します。これらの結論の多くは必ずしも直観的ではありませんが、いったん理解すれば、ワイヤレスセンサー・ネットワークを SmartMesh 製品に基づいて計画する方法に関するシステムレベルの決定を行う素養が十分に身に付きます。以下の分析では SmartMesh IP プラットフォームを使用しますが、この技術は SmartMesh WirelessHART ネットワークにも同様に当てはまり、同じスプレッドシートでカバーされます。

> ⚠ このアプリケーション・ノートに記載する定量的な結果は、古いバージョンの SmartMesh Power and Performance Estimator を使用して得られた値なので、最新バージョンを使用した場合は電流および待ち時間が若干異なることが予想されます。しかし、このアプリケーション・ノートで重要なのは定性的な結果であり、これは引き続き有効です。

24.2 問題：バッテリ寿命はどれくらいか？

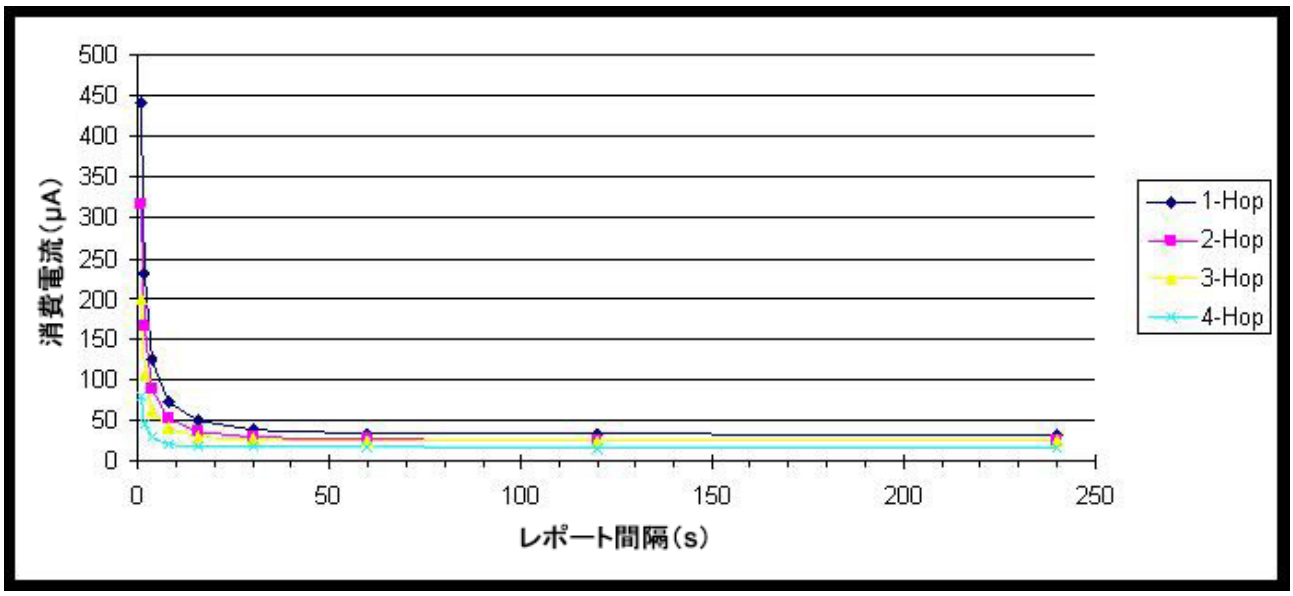
ワイヤレス・センサー・ネットワーク（WSN）の市場では、バッテリ寿命や電力消費が差別化の重要な機能です。SmartMesh 製品は、消費電力が最小の無線デバイスと最高のプロトコルを使用して無線デバイスのデューティ・サイクルを高くするので、バッテリ寿命のどのような比較でも勝利できる絶好の位置に付けています。とは言っても、デバイスを取り付けた瞬間には、デバイスのバッテリ寿命がどれくらいの長さになるか分らないのが事実です。このデバイスは他のデバイスのデータのルーティングを担当しないメッシュ内のリーフ・ノードになりますか？このデバイスは大量の負荷がかかったルータになりますか？ネットワーク内での再試行率によってデバイスの動作頻度は高くなりますか？これらの不確実性の累積効果は何ですか？このデバイスはバッテリ寿命を半減させますか？5 分の 1 ですか？100 分の 1 ですか？

24.3 Power and Performance Estimator

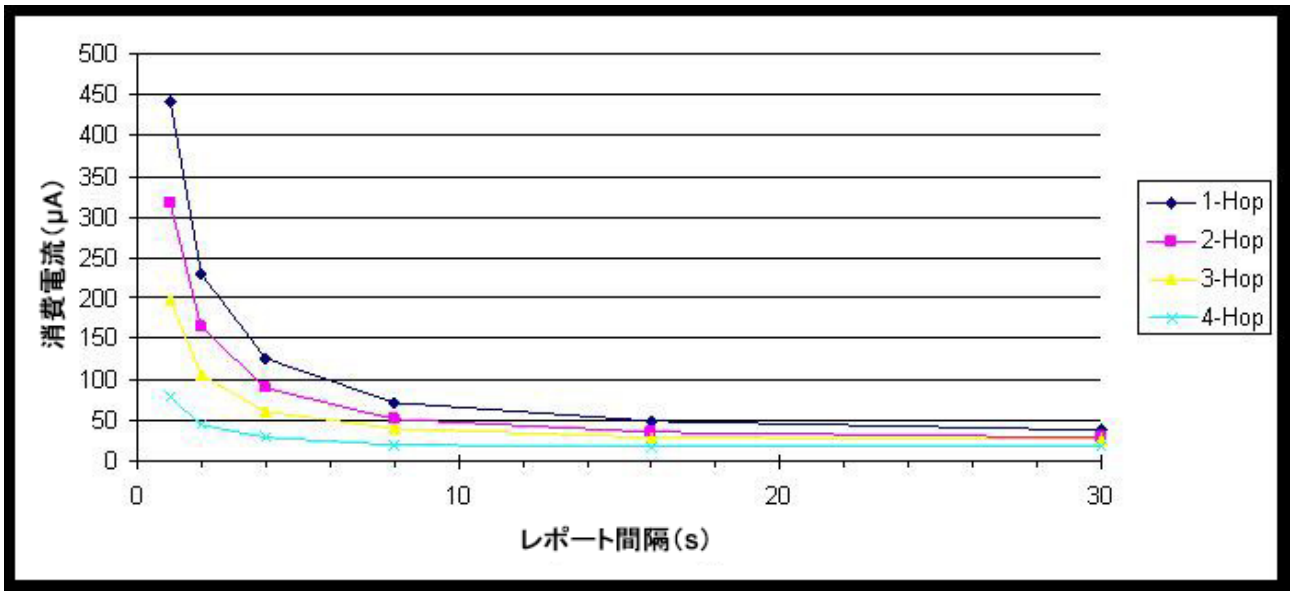
電力消費とバッテリ寿命に関連した不確実性の一部を取り除くために、弊社では SmartMesh Power and Performance Estimator を開発しました。この Excel スプレッドシートとやりとりして、指定できるネットワーク内で動作するモートの消費電力およびバッテリ寿命の推定値を求めることができます。SmartMesh Power and Performance Estimator は、消費電流とバッテリ寿命を推定するための概算ツールですが、感度の検討や相対的な比較を行うための優れたツールです。ここで導き出された結論は、これらの相対的な比較に基づいています。すべての例では、デフォルト値である 3.6V 電源と室温の 25℃を使用しています。

24.4 Q1：報告率が電力に及ぼす影響は？

スプレッドシートによる演習：ネットワークを定義して、すべてのモートの電力が可能な最小の消費量になるポイントを見つけるまで、報告率を低くします。4 ホップごとに 5 モートで計 20 モートの IP ネットワークを構築します。データのペイロード・サイズを 80 バイトで設定し、報告率を 1 パケット/秒に設定する予定です。その結果、1 ホップのモートは 441µA を消費しています。公称容量が 2400mA-hr の Tadiran TL4903 のようなバッテリを使用すると、7 ヵ月のバッテリ寿命に相当します。メッシュ内に子モートがない 4 ホップのモートのバッテリ寿命は約 3.1 年です。ここで、2 秒間隔、5 秒間隔、長時間の間隔で報告を繰り返します。結果は以下のとおりです。



データ報告率を 1 パケット/秒から 1 パケット/2 秒に低下させた場合、消費電流はほぼ半分に削減され、バッテリの寿命は実質的に 2 倍になりました。報告率が低い場合、これはあまり重要ではありません。センサーがデータを報告する頻度が 1 分に 1 回でも 4 分に 1 回でも、消費電流は平坦なままです。1 秒から 30 秒までの範囲を拡大すると、次のように表示されます。

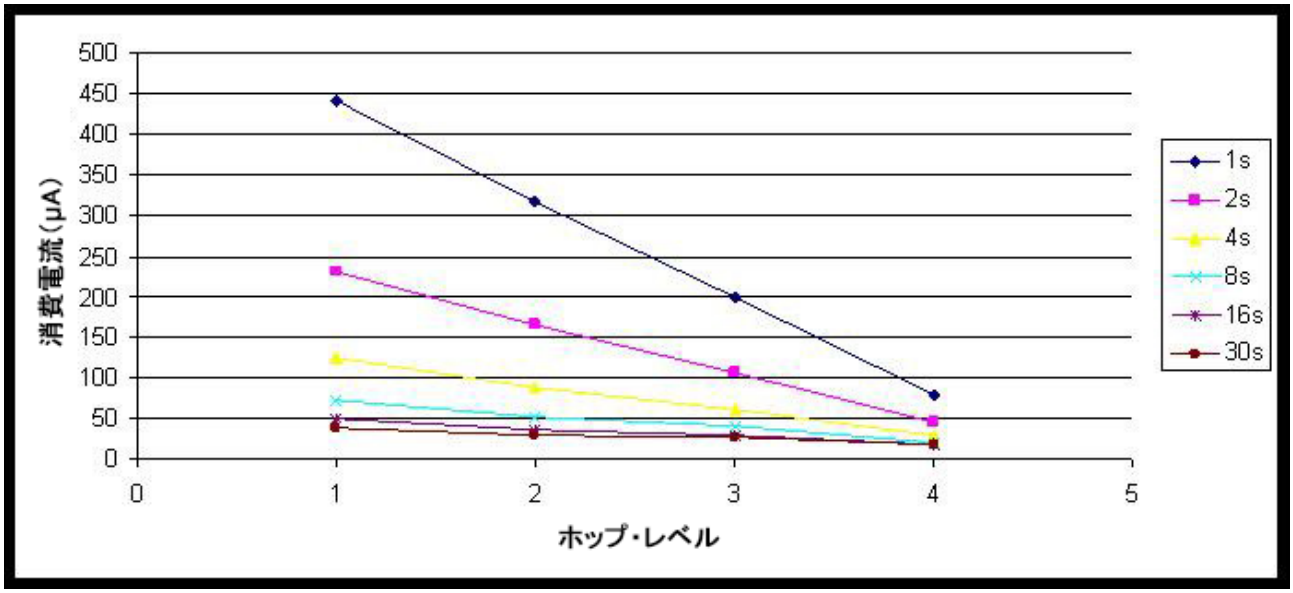


すべてのデバイスの消費電流が 100µA 未満のマルチホップ・メッシュを構築するのは、非常に簡単です。100µA では、TL4903 バッテリの場合は 2.5 年のバッテリ寿命に相当します。重要な結論は、高頻度の報告が必要な場合、バッテリ寿命を極めて長くするのは困難であるということです。バッテリの絶対最大寿命が必要な場合、データ報告頻度を 1 分に 1 回より低くするメリットはほとんどありません。

結論 1:それ以上低くしても意味がない最小報告率が存在します。SmartMesh ネットワークでは、ネットワークを介して時間同期を維持するために必要な一定の無線トラフィック量があります。センサーがまったくデータを送信しない場合、ネットワーク内のトラフィックはこの時間同期(「キープアライブ」)トラフィックによって決まります。これにより、ネットワーク内での可能な最小消費電力の下限が設定されます。センサーがデータを送信するときは常に、この時間同期トラフィックを送信する必要がない時間を表しています。

24.5 Q2:ルーティング・コストはどれくらいか?

スプレッドシートによる演習:上記と同じデータ設定を使用すると、ルーティングのコストを示すことができます。この 4 ホップ・メッシュでは、4 ホップ・モートがモート固有のデータを送信します。3 ホップ・モートは、モート固有のデータに加えて、メッシュ内の子モートからのデータを送信します。1 ホップ・モートは、ネットワーク内のすべてのトラフィックに加えて、1 ホップ・モート固有のトラフィックを転送する役割を果たします。データを様々な報告率で切り分けると、ルーティングのコストを説明することができます。

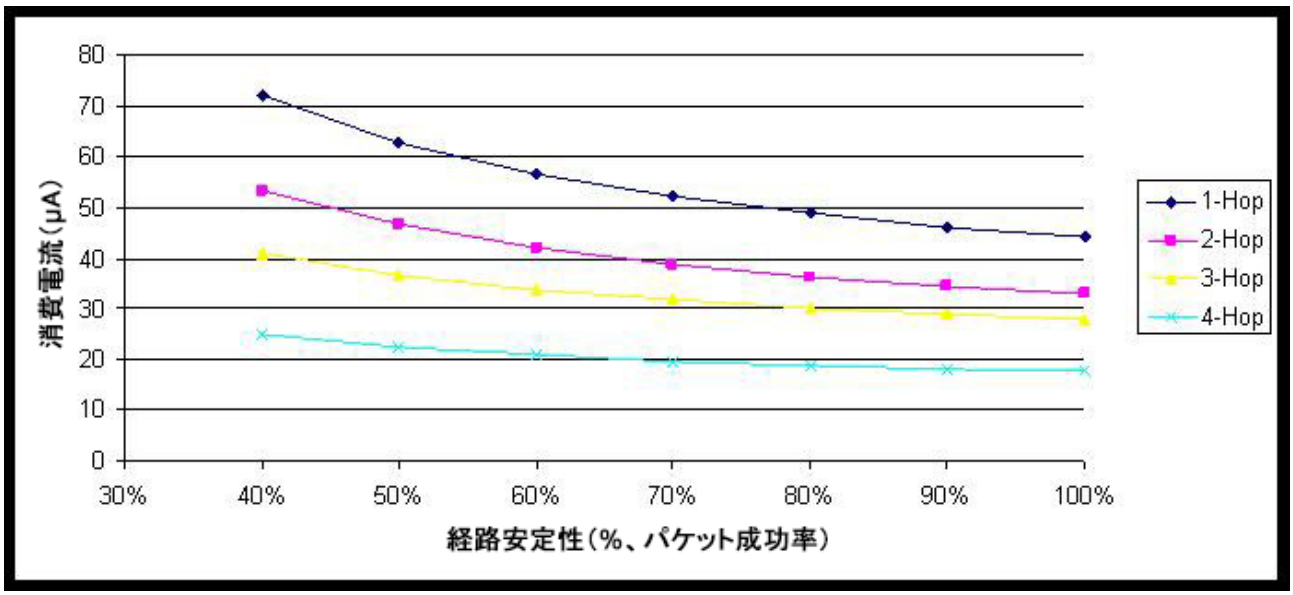


報告率が低い場合、ルーティング・データのコストは横ばいになります。4 ホップ・モートの消費電力は常に最小ですが、すべてのルーティングを行うデバイスの消費電流が必ずしも数倍になるとは限りません。適度な報告率では、消費電流特性が非常に平坦で適度に一様のバッテリ寿命を備えたネットワークを構築することが可能です。

結論 2:ある程度の負担がルータにかかりますが、特に報告率がキープアライブ間隔に近づくにつれて予想したほどの大きさにはならないと思われます。

24.6 Q3：再送信コストはどれくらいか？

スプレッドシートによる演習：スプレッドシートでの入力の 1 つは経路安定性です。これはパケット成功率、つまり「1 から再試行率を引いた値」を意味します。無線のプロトコルを最適化してビット・エラーおよびパケット・エラーを減らすことが、消費電力削減の鍵であると考える人もいます。これらのネットワークでは約 30%の再試行率は珍しくありませんが、これはバッテリの消耗速度が本来より 30%速いことを意味するのでしょうか？これを説明する方法は、20 デバイスで 4 ホップのネットワークの報告率を 16 秒間隔に維持することです。経路安定性を変化させます。結果は以下のとおりです。

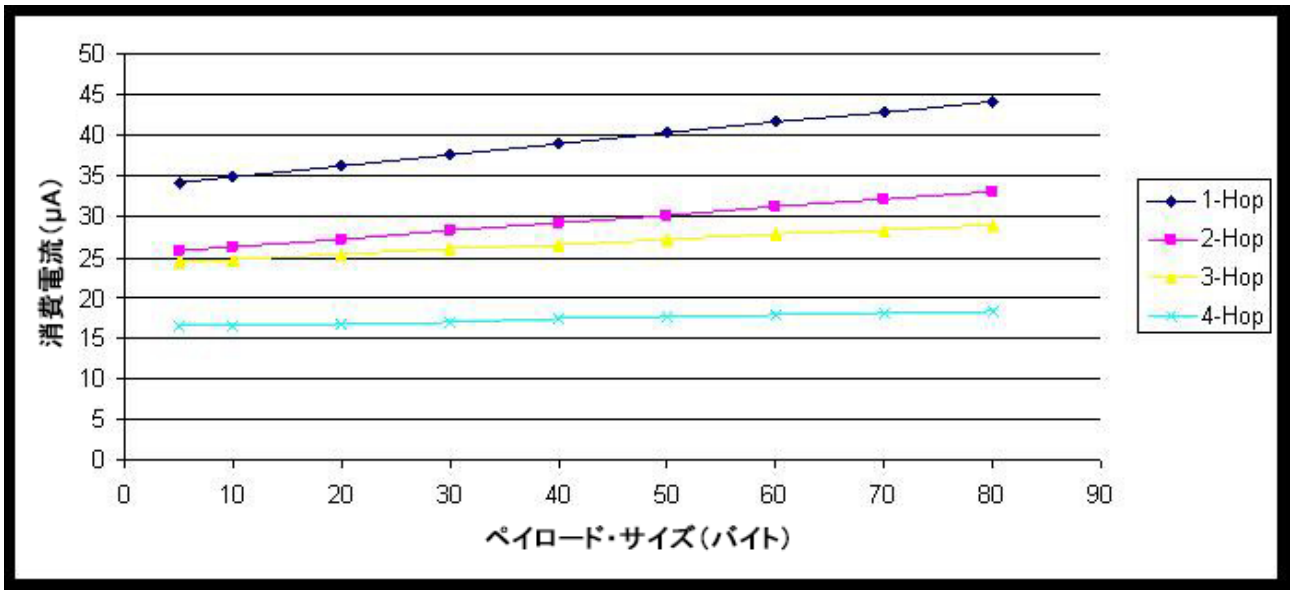


この適度な報告率では、パケット成功率が高いと消費電力が減少するのが事実です。パケット成功率が 40%であれば、消費電流は 2.5 倍になり、バッテリの寿命は半分に満たなくなると思う可能性がありますが、実際はそうではありません。必要な再試行回数が少ない強力な RF 経路が存在する方が望ましいことは確かですが、再試行の実行がバッテリ寿命にとって致命的であると考える必要はありません。SmartMesh マネージャは、経路安定性とホップ数の間のトレードオフを理解し、それに応じて最適化します。経路安定性が高い親より AP に近い親を選択した方が良い場合があります。こうするとネットワーク内での全再試行数は増加する可能性がありますが、各パケットが宛先に到達するために経由する必要があるホップ数は少なくて済みます。40%より低いパケット成功率でネットワークが十分に機能するとは予想していません。非常に低いパケット成功率では、経路が完全に失敗する境界にあることが分ります。この場合の問題は、メッシュからデバイスが完全に失われることです。

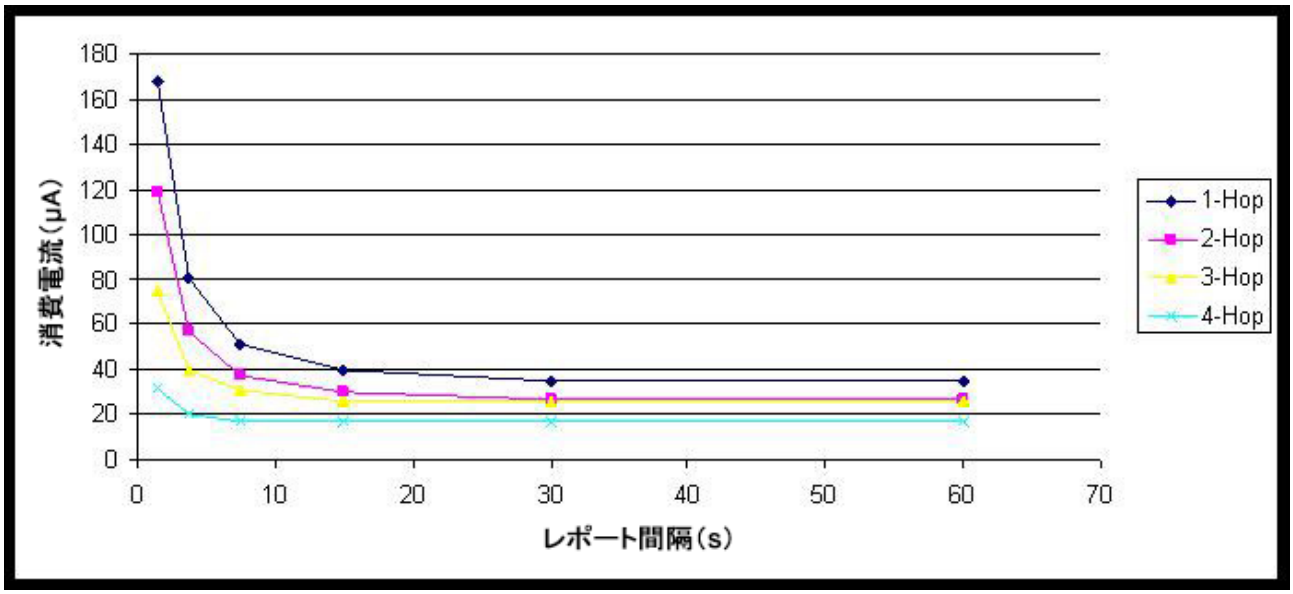
結論 3：再試行率は重要ですが、思ったほどではない可能性があります。

24.7 Q4：パケット集約で電力を節減できるか？

スプレッドシートによる演習：同じ 20 モート、4 ホップの深いネットワークで、1 回の報告間隔を 20 秒に選択し、ペイロード・サイズを 5 バイト～80 バイトの範囲で変化させます。結果は以下のとおりです。



より大きなデータ・ペイロードを送信すると消費電力が増加するのは確かですが、90 バイトのペイロード中のデータを 18 倍送信しても消費電流の増加は約 10%にすぎません。これを別の見方でとらえるため、すべてのセンサーが 1 分に 80 バイトを送信する場合を考えます。あるセンサー・アプリケーションは、80 バイトのペイロードを 1 分に 1 回送信します。別のセンサー・アプリケーションは、2 つの 40 バイト・ペイロードを 30 秒に 1 回送信します。3 番目のセンサー・アプリケーションは 20 バイトのペイロードを 15 秒ごとに送信し、4 番目のセンサー・アプリケーションは 10 バイトのペイロードを 7.5 秒ごとに送信します。5 番目のセンサー・アプリケーションは 5 バイトのペイロードを 3.75 秒ごとに送信します。6 番目のセンサー・アプリケーションは 2 バイトのペイロードを 1.5 秒ごとに送信します。結果は以下のとおりです。

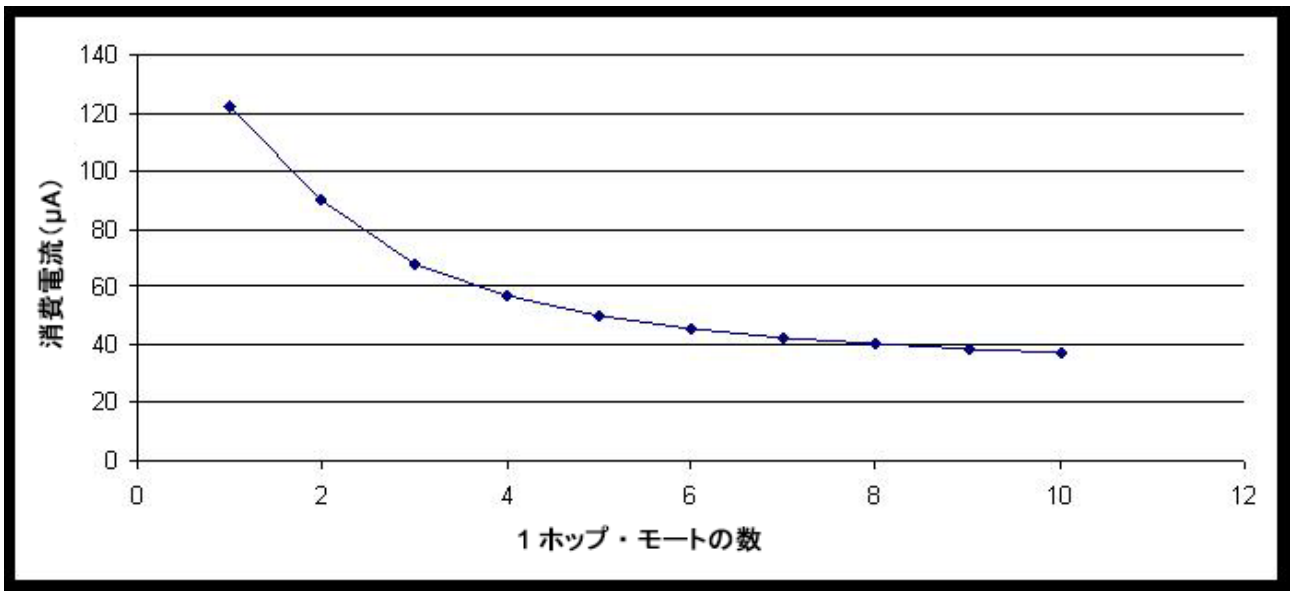


上記のすべてのアプリケーションは、完全に同じセンサー・スループットで送信しています。より大きなサイズのペイロードを低い頻度で送信する方がはるかに効率的なデータ転送方法です。トレードオフはもちろん待ち時間です。アプリケーションによっては、新しく読み取ることができる場合は古いデータは必要ありません。別のアプリケーションは、履歴データを使用して非常に優れた決定を行うことができます。消費電力は約 4 倍になりますが、空いているペイロード容量をアプリケーションが効率的に使用できる場合は、バッテリ寿命上の利点を得ることができます。

結論 4：小さなペイロードの送信回数を増やすよりも、大きなペイロードを少ない回数で送信するほうが効率的です。また、すべてのパケットには固定のオーバヘッドがあるので、大きなペイロードを送信しても、小さなペイロードの送信より大幅に時間がかかることはありません。

24.8 Q5：ネットワーク深さが電力に及ぼす影響は？

スプレッドシートによる演習：この同じ 4 ホップ・メッシュを再検討しますが、今度は 1 ホップ・モートの数を変化させます。ホップ 2～4 は 12 モートのセンサー・クラスタとみなし、1 ホップ・モートは、センサーからゲートウェイまでのギャップを埋めるために顧客が仕方なく購入している単なる中継器であるとみなします。すべてのセンサーが 15 秒に 1 回 80 バイトのペイロードを送信します。1 つの 1 ホップ・モートから始めて、1 ホップ・モートが 10 個になるまでモートを追加し続けます。最も厳しい場合の消費電力を 1 ホップ・モートの関数としてグラフ化します。結果は以下のとおりです。

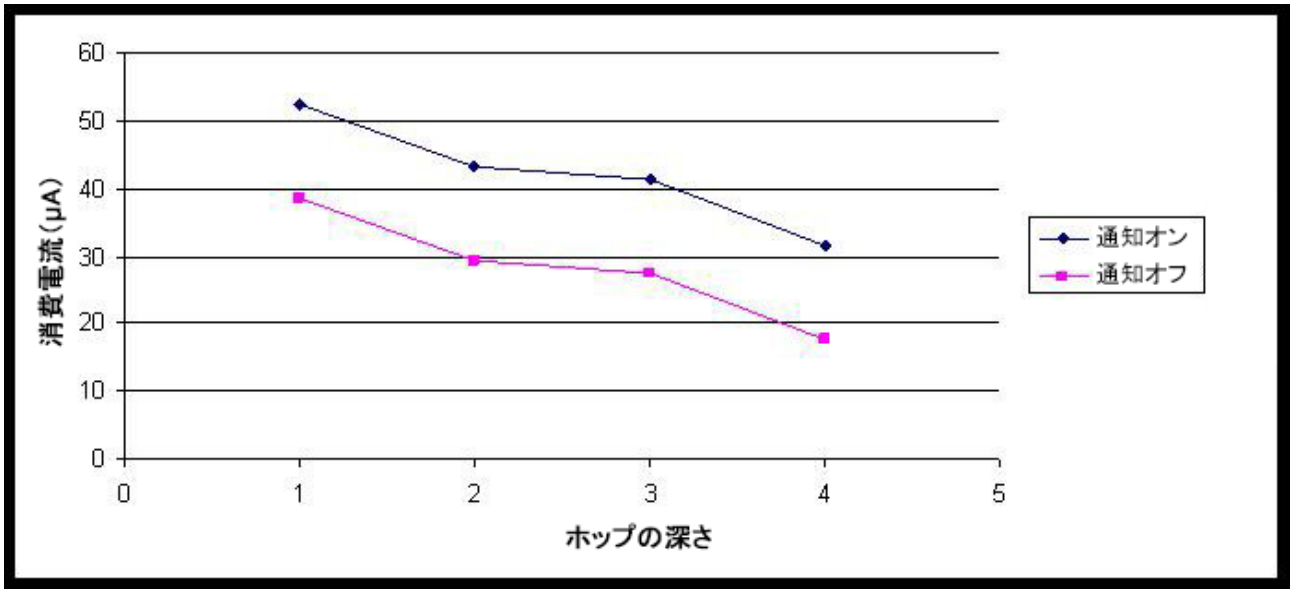


多くのシステムでは、「バッテリ寿命」の概念は、バッテリの消耗によって見えなくなるセンサーが**初めて**出た瞬間を意味します。そのときには、すべてのバッテリが交換されます。1 ホップのリングはすべてのトラフィックを転送する必要があるので、最初に駄目になったバッテリが 1 ホップのリングの中にあることはほぼ確実です。この期間を長くする最も簡単な方法は、多くの 1 ホップ・デバイスを用意することです。この効果に加えて、多くの 1 ホップ・デバイスを備えておくと、離脱デバイスに対する堅牢性が高くなります。ネットワークに 2 つの 1 ホップ・デバイスしか存在せず、1 つが取り除かれると、その最終ホップにはもはや良質なメッシュが存在するとはいえません。信頼性が低下する可能性があります。一般的に、トラフィックはマネージャに向かうすべてのホップ・レベルで増加します。各ホップ・レベルでは、そのホップ・レベルのモートがそれら固有のトラフィックを転送するのに加えて、子孫モートからのトラフィックをすべて転送しています。ゲートウェイに近い個々のホップ・レベルでは、消費電流の点から可能な最善の方法は、該当のホップ・レベルにあるすべてのデバイスが作業を均等に分担することです。該当のホップ・レベルにあるデバイス数を増やすほど、均等な分担作業量が小さくなり、これらのデバイスのいずれか 1 つが離脱した場合の不利益も小さくなります。

結論 5：できるだけ多くの 1 ホップ・モートを使用してください。任意のホップのリングに流れる電流は、そのホップのリングにあるモートの数に反比例します。

24.9 Q6：IP ネットワークでの通知を停止して電力を節減？

スプレッドシートによる演習：通知をオンにするための加算器があることに注意してください。IP ネットワークでは、通知はデフォルトでオンになっています。省電力化が望ましい場合は、アプリケーションが通知をオフにする必要があります。ただし、注意が必要です。通知がオフになっている間、新しいモートはどうしても参加することができません。したがって、アプリケーションによって通知をオフにする場合は、オンに戻すための仕組みが必要です。例えば、ネットワークを形成する場合、通知をオフにして、あるモートの電源を切ってから再度投入すると、通知をオンに戻さない限り、そのモートが再参加することはありません。モートが離脱するたびにアプリケーションが自動的に通知をオンにすれば、問題はありません。また、アプリケーションが通知をオンにするユーザ・インターフェースを備えている場合、新規デバイスの追加も容易に行うことができます。アプリケーション開発者が注意深ければ、通知を実質的に常時オフにすることができます。これは、モートごとに 13.8μA の節約に相当します。30 秒間隔で更新する 4 ホップ、20 モートのネットワークを見直してみると、以下の電力曲線が得られます。



31μA が 17.2μA に減少することにより、リーフ・ノードのバッテリ寿命は、Tadiran の単 3 電池（2160mA-hr）の場合で 14.5 年にすることが可能です。

結論 6：高速通知の有用性が低い場合はオフにしてください。ただし、注意が必要です。

25 アプリケーション・ノート：移動するモートに伴って予想されること

25.1 モートの移動

現在の SmartMesh 製品では、その親の範囲を越えて移動するモートは、リセットして新しい親でネットワークに再参加する必要があります。同様に、既存のネットワークの無線範囲内に到達する新しいモートがデータを送受信できるようになるには、その前にそのネットワークに参加する必要があります。これらの条件での正確な動作は、どの製品が使用されているかによって異なります。

ネットワークの周囲を絶え間なく移動し、1 つの無線範囲を超えて移動するモートは推奨しません。

25.2 SmartMesh WirelessHART

SmartMesh WirelessHART では、ネットワークがいったん構築段階から定常状態段階に移行すると、エネルギーを節約するためにモートでの通知が減少します。通知タイマは、モート上では 1.28 秒から 20 秒に長くなります。これは、定常状態時にモートのアドバタイザの範囲内だけに現われるモートは、同期をとるのに構築段階時よりも約 16 倍長くかかることを意味します。AP 通知の間隔は 0.16 秒から変わらないので、AP の範囲内に現われる新しいモートの参加動作は、2 つの状態のいずれでも変わりません。アプリケーションは、新しいモートが到着することを予想している場合、API を使用して通知の頻度を再度高くすることもできます。

移動するモートが既にネットワーク内にある場合、モートの親経路はリセット前に両方ともなくなっている必要があります。WirelessHART 標準での経路アラーム・タイマは 4 分です。また、こうした経路アラームを受け取ることが、モートが親経路を削除するきっかけになります。そのため、移動後のモートがリセットするまでには 4 分より少し長くかかると予想されます。いったんモートがリセットすると、マネージャはモートの以前の下りリンクによってモートに到達しようとします。いったんこの状態マシンがタイムアウトすると、それは最大 15 分に達することがあるので、離脱したモートを探索するために残りのモートでは通知が自動的に促進されます。その一方で、モートがその新しい場所で通知を受信した場合は、モートが離脱していることをマネージャが検出する前に、モートはネットワークに再参加できます。高速通知の期間中、新しいモートの構築時または探索時に、Eterna モートは余分に 16 μA を消費し、DN2510 ベースのモートは余分に 30μA を消費します。

参加までの予想時間の詳細については、SmartMesh WirelessHART User's Guide の Network Formation（ネットワーク形成）セクションを参照してください。

25.3 SmartMesh IP

SmartMesh IP では、アプリケーションが API を介して通知の動作を明示的に停止しない限り、通知は同じ期間維持されます。SmartMesh IP の各フレームの長さは約 2 秒です。モートの通知頻度は 1 フレームにつき 1 回であり、AP の通知頻度は 1 フレームにつき 4 回です。通知の動作が停止した場合、通知を行うモートはなくなります。つまり、新しいモートは参加できず、モートはネットワークに再参加できません。エネルギーの節約が不可欠で、マネージャの

介在なしで通知を制御するのに十分なノウハウをアプリケーションが備えている場合に限り、通知を停止することを推奨します。通知を停止すると、Eterna モートでは 14µA 節約できます。

移動するモートが既にネットワーク内にある場合、モートの親経路はリセット前に両方ともなくなっている必要があります。SmartMesh IP での経路アラーム・タイマは 1 分です。また、こうした経路アラームを受け取ることが、モートが親経路を削除するきっかけになります。そのため、移動後のモートがリセットするまでには 1 分より少し長くかかると予想されます。下りの照会を持つモートにマネージャが到達できないと、そのモートは**離脱**とマークされますが、通知の自動変更は行われません。その間に、移動したモートは、新しい通知を受信した場合は必ず、自由に再参加できます。

参加までの予想時間の詳細については、SmartMesh IP User's Guide の Network Formation（ネットワーク形成）セクションを参照してください。

25.4 SmartMesh WirelessHART と SmartMesh IP との違いのまとめ

この単純な比較では、80%の経路安定性と 5%のジョイン・デューティ・サイクルを想定しています。

パラメータ	WH	IP
リセットまでの時間	4 分	1 分
通知頻度を高めるまでの時間	15 分	非該当
定常状態の平均同期時間（1 ホップ）	60 秒	188 秒*
定常状態の平均同期時間（マルチホップ、3 つの隣接モート）	42 分	250 秒*

* SmartMesh IP のアプリケーションによって通知が明示的にオフになった場合、モートは同期できず、ネットワークに参加できません。

26 アプリケーション・ノート：ネットワーク間でのモートの移行

26.1 モートの移行

ネットワークに加えるモートの数をネットワークの存続期間にわたって徐々に増やします。ある一定のネットワーク・サイズでは、マネージャがそのモートまたは帯域幅の限界に達する場合があるので、いくつかのモートを新しいネットワークに移動することが望ましいか、移動することが必要な場合があります。本書では、顧客のアプリケーションがモートの機能を使用してこの処理を自動的に管理する方法について説明します。これについては SmartMesh IP 組込みマネージャ・ネットワークのコンテキストで説明しますが、その手順は SmartMesh IP VManager ネットワークおよび SmartMesh WirelessHART ネットワークにも同様に当てはまります。

26.2 手順

ネットワークを石油採掘場で配置します。新しい坑井が掘削されるにつれて、モートが追加されます。次に示す図では、ネットワーク A が限度容量に近づいており、2 番目のマネージャはネットワーク B を形成するために最新の採油所の近くに配置されます。最初はネットワーク B にわずか数モート（灰色の菱形）しかないので、ネットワーク A から数モート（白い菱形）を移動することが望まれます。ネットワーク A とネットワーク B のネットワーク ID は異なります。

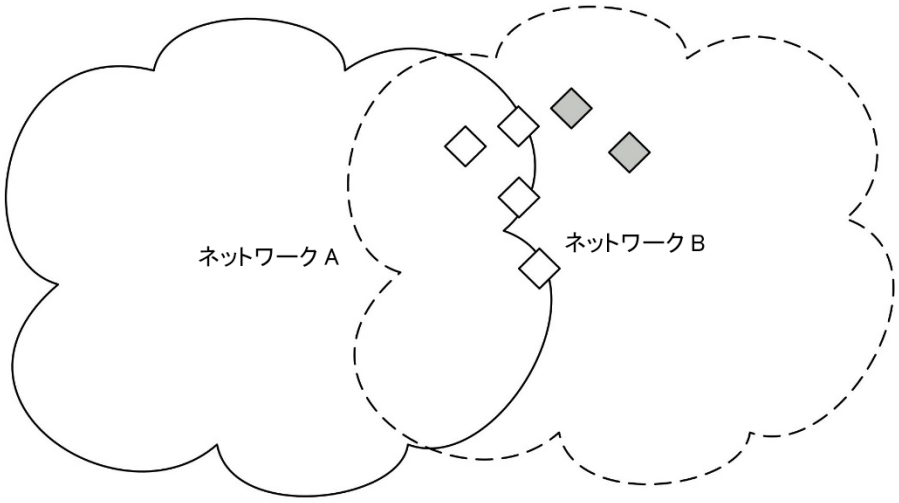


図 1 - ネットワーク A からネットワーク B へのモートの移動

このためには、**search** API を使用します。これにより、モートは、近辺にあるどのネットワークからの通知の受信も待機して、受信した通知の送信元ネットワークを報告するモードに入ります。

1. モートはネットワーク ID で事前に設定されます。これを「優先」ID と呼びます。これはモートの使用を開始する場合の通常の手順で、モートがその後移動すると予想されるかどうかは関係ありません。

2. センサー・アプリケーションは探索 API を使用して、受信した通知に関する情報の報告をモートが開始するよう動作します。モート ID、ネットワーク ID、および信号レベル（RSSI）が報告されます。これらの情報はセンサー・アプリケーションによって保管されます。
3. センサー・アプリケーションは探索のタイムアウトを設定します。ここでは、**joinDutyCycle** パラメータにより倍率が逆方向に変更されます。10%のデューティ・サイクルでは、タイムアウトは 2～3 分です。与えられたデューティ・サイクルでの探索が長いほど、近辺にあるすべてのネットワークを検出する確率は高くなりますが、この処理で消費されるエネルギーも高くなります。
4. タイムアウトの終了時に、優先ネットワーク（モートの現在のネットワーク ID）が十分な RSSI（>-85dBm）で受信されている場合、センサー・アプリケーションは **join** コマンドを出します。優先 ID が十分な RSSI で受信されない場合、センサー・アプリケーションは十分な RSSI を持つ別の検出済みネットワークを選択し、**networkId** パラメータをそのネットワークに設定して、join コマンドを発行する必要があります。
5. モートが優先 ID に数回参加しようとするが成功しない場合は、モートに適切な資格情報（後述の「セキュリティ」を参照）が存在しない可能性があります。また、センサー・アプリケーションは探索処理を再開する一方で、モートが参加できないこのネットワークを「ブラックリストに登録」する必要があります。

一般に、この方法を使うと、モートは、モートが備える適切なセキュリティ資格情報の対象となる利用可能なすべてのネットワークに参加できます。探索 API は、顧客のセンサー・アプリケーションが使用する参加ロジックの正常な部分（つまり、モート API を駆動しているすべてのロジック）として、あるいは顧客のアプリケーション・メッセージに応じて呼び出すことができます。上記のネットワーク A および B の場合には、ホスト側の顧客アプリケーションが白い菱形のモートの優先ネットワーク ID を B に変更した後、対象のモートをリセットします。その後、これらのモートは上記の手順に従ってネットワーク B に参加しようとしますが、実際にネットワーク B の範囲内に存在しない場合、ネットワーク A に安全に戻ることができます。

26.3 セキュリティ

⚠ ACL 項目が何千もある SmartMesh IP 組込みマネージャを使用するためには、外部 SRAM のインストールが必要であり、またマネージャのバージョンが 1.4.1 以降である必要があります。

モートは通知の受信に加えて、その参加鍵をマネージャが使用するものと一致させる必要があります。望ましいやり方は、モートごとに固有の参加鍵にすることですが、この参加鍵はマネージャのアクセス制御リスト（ACL）にあります。ネットワークに参加できるのは、マネージャの ACL に登録されているモートだけです。ネットワーク A および B の場合には、ネットワーク A から移動した白のモートをネットワーク B の ACL に追加してからリセットする必要があります。

27 アプリケーション・ノート：境界内データ待ち時間に関するネットワークの構成

27.1 プロビジョニングの増加

デフォルトの設定では、SmartMesh ネットワークは低消費電力で高信頼性動作を目的として設計されています。安定性の平均値が 70%を超え、4 ホップ以下の標準的なネットワークでは、デフォルトの設定で、パケットの約 90%が 1 回の報告間隔内に配信されることが分っています。例えば、モートがモートのセンサーから 10 秒ごとに 1 パケットをそれぞれ配信しているネットワークでは、これらのパケットの 90%では 10 秒以下の待ち時間になると期待します。**プロビジョニング係数**と呼ばれる設定値を調整することにより、ネットワーク内で使用される電力を増やし、より高い割合のパケットを所定の目標待ち時間より短時間で配信できます。

このアプリケーション・ノートでは、アプリケーション・ノート「プロビジョニング係数の変更によるマネージャ・スループット」の増加での説明内容と反対のことを行います。

27.2 制限事項

このアプリケーション・ノートで説明する設定は、センサーがほぼ同じ割合で報告するネットワークに対して適用します。例えば、あるモートのセンサーが 1 秒間隔で報告し、それ以外のすべてのモートが 30 秒間隔で報告するネットワークでは、これらの設定が必ずしも機能するとは限りません。この種のシナリオの場合は、上りバックボーンを使用してください。IP ネットワークの場合は、アプリケーション・ノート通電バックボーンを使用した待ち時間の改善を参照してください。WirelessHART ネットワークの場合、この機能はバージョン 4.1.x 以上のマネージャで使用できます。

安定性の平均値が低い（70%未満の）ネットワークでは、待ち時間が長くなります。これらの場合には、本書での推奨設定値より高いプロビジョニング係数が必要です。最後に、本書での設定が当てはまるのは 4 ホップ以下のネットワークです。より深いネットワークの待ち時間はほぼ直線的に増加するので、例えば 8 ホップのネットワークは、本書で説明したプロビジョニング・レベルを 2 倍にすることにより、同じ待ち時間目標値を満たします。

27.3 プロビジョニング

すべての SmartMesh ネットワークでのデフォルトのプロビジョニングは 3 倍です。これは、すべてのモートには、予想されるパケット伝送ごとに 3 つ以上の上りリンクがあることを意味します。このレベルのプロビジョニングにすると、低消費電力（リンク数を減らして実現）および高い信頼性（リンク数を増やして実現）の正しいバランスをネットワークの範囲全体に対して達成できることが経験的に分っています。したがって、あるモートがローカル生成パケットと転送パケットの両方を含むパケットを 1 秒に 1 回送信している場合、このモートには 1 秒当たり 3 つの送信リンクが与えられます。

また、各モートがその親との同期を確実に維持できるようになる最小のリンク数があります。一部のモートでは、モートがデータ・パケットを単独で送信する必要があるリンクの数よりも、この最小数の方が実際には大きくなっています。

例えば、SmartMesh WirelessHART ネットワーク内の各モートは、10.24 秒当たり少なくとも 4 つの送信リンクを獲得します。モートのデータ要件が 60 秒当たり 1 パケットである場合、プロビジョニングが 3 倍とは、モートが 20 秒ごとに 1 つの送信リンクを必要とすることを意味します。4/10.24 という最小リンク数は、必要とされる 1/20 よりもはるかに大きい値です。

27.4 プロビジョニングの変更

経験則として、3 倍のプロビジョニングでは、パケットの 90%がマネージャに「時間どおりに」到達します。これは、10 秒間隔で報告するモートのパケットの 90%は、その待ち時間が 10 秒未満であり、7 秒間隔で報告するモートのパケットの 90%は、その待ち時間が 7 秒未満であることを意味します。すべての顧客が時間どおりのパケット配信を望むとは限らないことに注意してください。多くの場合、その目的は、合理的だが要求は厳しくない目標の期間内に、生成されたすべてのパケットをとにかく配信することです。

時間どおりの配信率を 99%まで高めるには、プロビジョニングを 6 倍に変更します。

- SmartMesh WirelessHART：`dcc.ini` に `TOP_LINK_OVRSUBSCR = 6.0` の 1 行を追加
- SmartMesh IP 組込みマネージャ：マネージャ CLI で、`set config bwmult 600` と入力
- SmartMesh IP VManager：マネージャ CLI で、`su becareful` と入力し、
  次に `config seti BWMULT=600` と入力

この変更を行うと、ネットワークの全スループット（単位：パケット/秒）が半分になることに注意してください。デフォルトの設定で 25 パケット/秒をサポートできるネットワークは、6 倍のプロビジョニングでは 12.5 パケット/秒しかサポートできません。

時間どおりの配信率を 99.9%まで高めるには、プロビジョニングを 9 倍に変更します。

- SmartMesh WirelessHART：`dcc.ini` に `TOP_LINK_OVRSUBSCR = 9.0` の 1 行を追加
- SmartMesh IP 組込みマネージャ：マネージャ CLI で、`set config bwmult 900` と入力
- SmartMesh IP VManager：マネージャ CLI で、`su becareful` と入力し、
  次に `config seti BWMULT=900` と入力

デフォルトの設定で 25 パケット/秒をサポートできるネットワークは、6 倍のプロビジョニングでは 8.3 パケット/秒しかサポートできません。

27.5 電力の増加

プロビジョニングを追加してもネットワーク内での送信パケット数は多くなりません。同じパケット数を送信するリンクの頻度が高くなるだけです。追加の送信リンクは電力を消費しないので、プロビジョニングが変更された場合、リーフ・ノードの電源要件が追加されることはありません。使用中のルータ・ノードは、いくつかの受信リンクを追加して追加の送信リンクと組み合わせるので、これらのノードが最も影響を受けます。この場合もやはり、経験則として、使用中のルータの消費電力は、プロビジョニングを 6 倍に増やすと 10～15%増加し、プロビジョニングを 9 倍に増やすと 20～30%増加します。

低トラフィックのネットワーク（例：報告間隔が 60 秒）の場合は、プロビジョニングを 9 倍に増やしても、ネットワーク内のリンク数やモートの消費電力が実際には増加しない場合があります。こうした場合には、リンクの最小数が、時間どおりの配信率 99.9%を確保するのに既に十分な値になっています。

28 アプリケーション・ノート：ネットワークの共存

28.1 重複ネットワーク

すべての SmartMesh ネットワークは、同じ 2.4GHz の産業、科学、医療（ISM）無免許無線バンドで動作します。ここで言う無免許とは、現地での電力制限に従えば、この帯域（バンド）で誰でも自由に動作させることができ、テレビや携帯電話の帯域では必要になる免許が不要という意味です。この帯域が選ばれたのは、この帯域が世界中で利用可能であり、IEEE 802.15.4 PHY 標準無線規格が存在することが理由です。したがって、多くの相互運用可能な無線機が市販されています。しかし、誰もがこの帯域を使用できるので、他の製品で混雑しています。802.15.4 b/g/n Wi-Fi、Bluetooth、コードレス電話機、ZigBee は、すべて同じ帯域で動作します。実際の配置では、時刻 T0 でチャンネル X に存在する干渉物（またはマルチパス・フェージング環境）が、時刻 T1 ではチャンネル Y に存在しないであろうという考えのもとに、SmartMesh ネットワークは、チャンネルを飛び越すことにより潜在的な干渉の周囲をくぐり抜けて送信する十分な余裕をもって設計されています。すべてのチャンネルにわたる平均的な性能に頼ることにより、いずれか 1 つのチャンネルがうまくいくことに頼るより結果は良好になります。他のプロトコルは 1 チャンネル（または数組のチャンネル）に留まるか、帯域全体を素早く飛び越える（Bluetooth）ので、近辺にある他の SmartMesh ネットワークが最も有害な干渉物になる可能性があります。非同期ネットワークが共通のスケジュールを共有する場合は衝突が続く可能性があるからです。本書では、重複ネットワークによる干渉を乗り越えるために実装された機能について説明します。

28.2 単一ネットワークでの衝突

SmartMesh マネージャは、単一ネットワークでの伝送間での衝突を防止するように動作を予定に入れます。この規則の例外は、3 つの場所に出現する「共有」リンクです。共有リンクは、参加するモートが参加先の親と連絡をとるために使用します。最悪の場合でも、モートがネットワーク内で稼働する前に、この時間中の衝突によってモートはリセット状態になります。SmartMesh IP では、通電バックボーンは共有リンクを使用して待ち時間を短縮します。これらのネットワークには、データ要件をすべて満たすのに十分な専用（つまり、非共有）帯域幅が引き続き存在するので、この場合は衝突があっても待ち時間の短縮はごくわずかです。SmartMesh WirelessHART ネットワークでは、モートはリンクを共有してネットワーク検出パケットを送信します。マネージャはこれらのパケットの衝突確率を低くするのに十分な長さをタイマに割り当てます。パケットが衝突した場合でも、検出情報は後で安全に収集されます。

通常のネットワーク・トラフィックは専用チャンネルのオフセット上で予定されるので、上記の特定の衝突シナリオでは、これらのトラフィックと衝突することはありません。また、単一ネットワーク内の同じチャンネル上で同時に 2 つの伝送が実行されることもありません。

28.3 周期的な衝突の回避

2 つのネットワークが同じ無線空間内にあり、両ネットワークとも以下の性質を持っているとします。

- 現在の絶対スロット番号（ASN）が同じ
- チャンネル・リスト（例：デフォルトのチャンネルのブラックリスト）が同じ

- すべてのスロットフレームの長さが同じ
- 同じタイムスロットのトランザクションのオフセットが同じ

この場合、ネットワークは両方とも同じチャンネル周波数で同時に送信するので、両方のメッセージが互いに干渉し、結果としてどちらも正常に送信されない可能性があります。該当するパケットは後で再試行される必要があります。ちょうど 1 スロットフレーム後に同じシナリオが繰り返され、これが繰り返され続けます。重要なことは、ネットワークが両方とも同じ周期で動作しているので、1 回発生した衝突は永続的に発生する可能性が高いということです。

ネットワークは、同じチャンネル・ホッピング・パターンに従う場合、必ずしも ASN を一致させて互いに干渉させるまでもありません。明示的に同期されないネットワークは、結局ドリフトします。また、いくつかの衝突が重複する期間があることが分ります。これらの期間は、相対的なドリフトによってタイムスロットの長さが経過するまでに要する時間を表しますが、数分から数時間続くことがあります。モートへの伝送が重複ネットワーク内での伝送パターンとすべて一致する場合、このモートはその親と十分長い時間通信することができずに終わり、通信が途切れてリセットする可能性があります。伝送が時折成功すれば、モートはネットワーク内にとどまります。モートが離脱するのは、すべてのデータ・パケットおよびキープアライブ・パケットが絶え間なく上書きされるときだけです。

2 つのネットワーク間で重複する量は非常に少量です。各ネットワークは単一のアクセス・ポイント（AP）ノードによってデータを収集します。また、各 AP が受信または送信できるのは、各タイムスロット内の 1 チャンネル上に限られます。SmartMesh ネットワークで利用可能な 15 チャンネルと比較した場合、ネットワーク内で最も混雑した場所での実際の伝送が、最大でタイムスロット全体の 1/15 に収まることが期待されます。更に、タイムスロットの全長がこの伝送で満たされることはないので、全帯域幅に占める割合はより低くなります。このことは、周期的かつ持続的な衝突を回避できる場合には、モートが同じ無線空間で安全に共存する余地が十分あることを示しています。衝突が発生するが、発生頻度が一定ではなく不規則である場合、これは両方のネットワーク内の経路安定性の全体的な減少として現れ、重大な問題が発生することはありません。

SmartMesh ネットワークは、IP 製品群と WirelessHART 製品群とでは、周期的な衝突を別の方法で回避します。

28.3.1 SmartMesh WirelessHART

各 SmartMesh WirelessHART ネットワークには、固定長の同じスロットフレームがあります。上りスロットフレームは 1024 スロット、下りスロットフレームは 256 スロットであり、通知スロットフレームは 128 スロットです。これらのスロットフレーム長は変更できないので、周期性を回避するために異なる仕組みが必要です。

上りの通信は、各モートに最低 4 つの送信リンクを与えることによってランダム化されます。これらのリンクのオフセットは完全にランダムに選ばれ、選ばれたタイムスロットには上りリンクごとに若干のジッタがあります。したがって、ネットワーク A からの 4 つのリンクのいずれかがネットワーク B と衝突する確率は、15^3 分の 1 未満、つまり 1:3375。

下りの通信をランダム化するには、ブロードキャストとマルチキャストの両方のリンクを AP に与え、モートの宛先ごとに 2 つの送信元経路を使用します。下流のモートに達する最初の試みが失敗した場合は、異なるタイムスロットおよび比較的不規則なタイムスロット上の最初のホップ伝送により再試行されます。また、マルチホップ経路に沿った後続の各伝送に対しても同じことが適用されます。

通知とネットワーク検出の伝送は、割り当てられたスロットごとにではなく、タイマを基準にして行われます。これは、これらの伝送が完全な周期性では決して行われず、隣接するネットワーク内で持続的な問題が発生しないことを意味します。

28.3.2 SmartMesh IP 組込みマネージャ

SmartMesh IP 組込みマネージャはメモリに制約があるので、各モートにいくつかのリンクを割り当てることができません。この制限を乗り越えるため、短いスロットフレームに少ないリンク数を割り当てますが、こうすると衝突が持続的に発生する確率が高くなります。ただし、IP ネットワークの基本のスロットフレーム長は 1 種類なので、これはランダム化するのが簡単です。ネットワークの起動時に、マネージャは上流、下流、および通知のすべての動作で使用するスロットフレーム長を 256～284 タイムスロットの範囲でランダムに選択します。

IP ネットワークでは、上りリンクのデフォルトの最小数は 2 です。ネットワーク内にあまり多くのモートがなく、消費電力が大きな問題ではない場合は、「ini」（初期）設定でこのパラメータの値を大きくして、持続的な衝突に対する耐性を高めることができます。

下りの通信をランダム化するには、ブロードキャストとマルチキャストの両方のリンクを AP に与え、モートの宛先ごとに 2 つの送信元経路を使用します。下流のモートに達する最初の試みが失敗した場合は、異なるタイムスロットおよび比較的不規則なタイムスロット上の最初のホップ伝送により再試行されます。また、マルチホップ経路に沿った後続の各伝送に対しても同じことが適用されます。

28.3.3 SmartMesh IP VManager

SmartMesh IP VManager は小規模なネットワークでは固定スロットフレーム・サイズを使用します。デフォルト構成では、上りと下りのスロットフレームはどちらも 512 スロット長で、1 つの経路の双方向に複数のリンクが割り当てられるので、持続的な衝突が生じる可能性が低くなります。ネットワーク・サイズが大きくなると、VManager で複数のデバイスにセルを再割り当てする必要が生じるので、モートは上りのリンク不良を監視して、この情報をマネージャに報告します。数千モート規模までネットワークが拡大すると、VManager は通知フレームと検出フレームをランダム化します。

28.3.4 IP - WirelessHART の混在環境

IP ソリューションと WirelessHART ソリューションは、それぞれ異なるタイムスロット長（7.25 ミリ秒および 10 ミリ秒）で動作します。このことだけで、両者を同じ無線空間で動作させても安全であり、両者間に持続的な衝突が生じる恐れはありません。

28.4 クリア・チャンネル評価

すべての SmartMesh 製品はオプションのクリア・チャンネル評価（CCA）機能を備えています。この機能を有効化すると、ネットワーク内のすべてのデバイスは伝送前の短期間に受信するようになり、別のデバイスの伝送を偶然受信した場合は、予定していた伝送を中止します。これは隣接する非同期ネットワークの共存を目的としています。前述したように、衝突を局所的に回避しているので、これはネットワーク内では効果がありません。CCA の使用はデフォル

トではオフです。IEEE 802.15.4 で定義されているように、送信を中止するための閾値は低いので、正常に動作できるネットワークは、強度が中程度の広帯域干渉物によって完全に遮断される場合があります。隣接する別の 802.15.4 ネットワーク・オペレータから依頼があり、更に彼らが共存の問題を経験している場合のみ、CCA の使用を検討することを推奨します。

28.5 実験結果

弊社のテスト研究所では、絶え間なく動作する約 900 のモートを備えた SmartMesh ネットワークが日常的に約 40 あります。ピーク・トラフィックの時間帯に、弊社ビル内で利用可能な各チャンネルで 20 パケット/秒以上を取り込みました。この環境は顧客の配置が直面したどんなことよりも難易度が高いと考えられ、衝突が原因でモートがリセットされることはめったにありません。

弊社では、弊社の小さな研究所で動作する 1000 のモートを 8x125 モートの WirelessHART ネットワークに分割して専用のテストを実行しました。このネットワークでのピーク時の総トラフィックは 181 パケット/秒で、ネットワークは 99.9%より高い信頼性を維持していました。モートが同期を維持していたがデータを送信しなかった低トラフィック設定と比較すると、経路安定性は 80%から 68%に低下しました。全体的に見て、ネットワークの待ち時間と消費電力は増加しましたが、深刻な問題にはなりませんでした。

29 アプリケーション・ノート：ジョイン・デューティ・サイクルの選択方法

このアプリケーション・ノートでは、モートの**ジョイン・デューティ・サイクル**を設定する方法の決定について段階的に説明します。この決定は、通常、モートとやりとりするセンサー・アプリケーションによって行われます。

29.1 背景 - ジョイン・デューティ・サイクルとは？

モートの **join** API コマンドを送信すると、モートは参加するネットワークを探索し始めます。探索とは、モートが 1 チャンネル上でしばらくの間受信を待機し、次にしばらくの間スリープ状態になった後、別のチャンネル上で受信待機を再開することを意味します。全期間（受信待機時間 + スリープ時間）の長さは 3～4 秒です（SmartMesh ファミリにより異なります）。ジョイン・デューティ・サイクルは、0～255 の範囲内のいずれかの値に設定できる 1 バイトのフィールドで、この期間のどの部分を受信待機に費やすかを設定できます。モートが受信を待機する時間の割合が大きいほど、そのモートが通知を受信してネットワークに参加する速度は速くなりますが、消費電流の平均値は大きくなります。例えば、モートの探索デューティ・サイクルを 255 に設定すると、モートは常に受信を待機し、数秒ごとにチャンネルを変更します。ジョイン・デューティ・サイクルを 26 に設定すると、モートは約 10%（26/255）の時間を受信待機に費やし、約 90%の時間をスリープ状態に費やします。こうすると消費電力は非常に少なくなりますが、通知の受信速度も大幅に低下します。したがって、ジョイン・デューティ・サイクルの設定は、平均電力と探索に費やす時間のトレードオフをもたらします。モートがネットワークに同期する期待時間は、ジョイン・デューティ・サイクルとネットワーク・トポロジ（通知を送信している隣接モートの数および送信率を介して表現）の関数です。

同期時間 = デバイス当たりの通知送信率 * チャンネル数 /（通知送信デバイス数 * 経路安定性 * 受信モートのジョイン・デューティ・サイクル）

SmartMesh スタータ・キット（DC9000、DC9021、DC9007）では、モートは**マスタ・モード**で動作します。モートは独力で参加し、外部プロセッサからのコマンドは使用しません。このモードは、通常はデモンストレーション用に使用されます。対照的に、最も現実的なアプリケーションはモートを**スレーブ・モード**で動作させます。このモードでは、モートが参加するようモートに指示する必要があります。join コマンドと共に、探索に使用するデューティ・サイクルをモートに指示する必要があります。本書では、いくつかのネットワーク・シナリオでの「十分に速い」ネットワーク形成時間と平均電力のトレードオフについて明確に説明します。SmartMesh WirelessHART では、リセットした場合や電源を入れ直した場合、モートはデフォルトに戻るので、デフォルト値以外の値が望ましい場合は、起動のたびに設定する必要があります。SmartMesh IP では、ジョイン・デューティ・サイクルはリセットや電源の入れ直しがあっても維持されるので、設定する必要があるのは 1 回だけです。

29.2 使用すべきジョイン・デューティ・サイクルは？

ジョイン・デューティ・サイクルを設定する一般的な方法は 4 つあります。

1. 通電デバイスの場合：単純に 100%に設定します。この設定の利点は、参加までの時間が常に最短であることです。隣接モートを検出できない場所にモートを置くと、そのデバイスは探索に約 5mA を消費しますが、通知を受信することはありません。
2. スカベンジャ・デバイスの場合：余裕のある電力レベルに設定します。スカベンジャ・デバイスが保証できる電流の大きさは限られています。その大きさに適合する探索デューティ・サイクルを設定します。例えば、連続して 200µA 供給できるスカベンジャをモートが備えており、モートが受信時に 5mA を消費する場合は、探索デューティ・サイクルを 10（10/255 = 4%）以下に設定する必要があることが分っています。モートの参加速度はデューティ・サイクルが高い場合より低下しますが、モートは探索の必要がある限り、予想電力量より低い値にとどまります。
3. バッテリ寿命の目標値と一致するように設定します。デバイスに厳密なバッテリ寿命の目標値がある場合、このデバイスには規定の平均予想電流量が存在し、上記 2 の場合のようにジョイン・デューティ・サイクルを計算できます。例えば、2000mAhr のバッテリがあり、目標の寿命が 10,000 時間である場合は、200µA を流す余裕があります。このように、デバイスはそれがネットワーク内にあるかどうかに関係なく、バッテリ寿命の目標値を満たします。これは、「立ち往生状態の」位置に残されることがよくあるが、ネットワークが存在するようになったら必ず参加すると予想されるデバイスに対する方法でもあります。
4. 速度と電力の点で「十分な」値を選ぶようにします。自分自身のためにテストした場合は、密集したネットワーク（ほとんどのモートが 8 つ以上の隣接モートを検出可能なネットワーク）を完全に形成するための総所要時間は、ジョイン・デューティ・サイクルとの関連がごくわずかであることが分ります。100 モートのネットワークを形成するのに探索デューティ・サイクルが 100%のとき 30 分かかる場合、50%でもわずか 32 分、25%でも 35 分しかかからない可能性があります。探索デューティ・サイクルを比較的低い値に設定することによって、速度はわずかに低下するものの、「立ち往生状態の」デバイスで大量の消費電力を節減できます。設定値が低すぎて 10%をかなり下回る場合は、事態が大幅に減速する可能性があります。限界では、探索デューティ・サイクルを 0（0.2%）にすることで、参加速度は極めて低くなり、平均消費電流は 10µA 未満になります。デバイスが非通電状態で、スカベンジャではなく、立ち往生状態のままになっている可能性も低い場合は、25%が優れたスタート・ラインです。設置担当者にとって事態の進行が十分に速いことを確認し、担当者のフィードバックに基づいてこの値を調整することを検討してください。

29.3 センサー・アプリケーションの状態マシン

SmartMesh IP モートはジョイン・デューティ・サイクルの設定を持続するので、通常この設定は製造時またはデバイスの最初の運用開始時に行われます。

SmartMesh WirelessHART では、ジョイン・デューティ・サイクルが持続されないので、アプリケーションに対して以下の規則の適用を検討してください。

規則 1：ソフトウェアが起動イベントを常に取得できる状況にしておく。

モートがリセットすると必ず、このイベントはモートによって送信されます。モートの電源が入れ直される場合、モートは起動イベントを送信します。モートはネットワークから離脱すると、起動イベントを送信します。モートはリセット・コマンドを無線で受信すると、起動イベントを送信します。最も単純な状態マシンは、起動イベントを受信するたびに、ジョイン・デューティ・サイクルを設定します。

規則 2:ソフトウェアは、join コマンドを送信する前に、ジョイン・デューティ・サイクルを設定できるよう常に準備が必要である。

デフォルトのジョイン・デューティ・サイクルを望んでいることが判明した場合でも、そのデフォルト値は将来変更される可能性があります。使用する値は常に書き留めておいてください(最善の実践例は、現在の値を読み取り、その値が目的の値と異なる場合は新しい値に書き換えることです)。

29.4 まとめ

繰り返すには、以下の手順に従います。

- ステップ 1:起動時の定型作業で、モートの API コマンドを使用してジョイン・デューティ・サイクルを設定します。join コマンドを送信するときは常にこのコマンドを呼び出すようにしてください。
- ステップ 2:プレースホルダとして、値 64(25%)を使用します。
- ステップ 3:約 5mA を常に消費するのが気にならない場合は、代わりに 255(100%)を設定します。
- ステップ 4:デバイスの予想電力量が厳密な場合は、その値を使用して、余裕のあるジョイン・デューティ・サイクルを計算します。
- ステップ 5:値を実験的にテストして、調整が必要かどうかを決めます。

30 アプリケーション・ノート：SmartMesh のセキュリティ

30.1 概要

無線パケットは、理論的には誰でも盗聴できる空気中を通って送信されます。実際に、802.15.4 の周波数範囲（2.4GHz の ISM バンド）内の全 16 チャンネルを同時にモニタできる、いわゆる**パケット・スニファ**があるので、チャンネル・ホッピング単独では外部のリスナからデータを保護するには不十分です。セキュリティ・プロトコルは、すべてのパケットの未加工ビットを受信するリスナがどの情報も解読できないように設計されている必要があります。弊社の全製品は、標準製品の一環として同じセキュリティ機能を備えています。弊社は、お客様が認識しているかどうかに関係なく、すべてのお客様が安全なネットワークを必要としていると考えており、SmartMesh 製品は常にセキュアな通信を使用するように構築されています。セキュリティ機能の唯一のマイナス面は、各パケット伝送に少量のオーバヘッドが追加されることですが、これによって生じる消費電力の増加分はごくわずかです。Dust 製品に実装されているセキュリティ基準は産業界の最良実施例であり、Dust の実装は徹底していて、斬新ではありません（安心です）。

30.2 目標

安全なワイヤレス・ネットワークには以下の特性が必須です。

- **メッセージ整合性**―宛先で受信されたデータは、伝送中に改変されていた場合、受け付けられないようにする必要があります。これは、悪意のあるルータが存在する場合でも、パケットが送信元から宛先まで多くのホップを通過する場合でも維持される必要がある終端間特性です。これは**認証**とも呼ばれます。会話している両者が気が付かないうちに第三者と通信しているという**中間者攻撃**を防ぐために、送信者の身元を確認することを意図しているからです。
- **アクセス制御**―モートは認証済みモートからのデータだけを受け付ける必要があります。これは終端間特性ですが、リンク層による当然の帰結でもあります。無許可のモートからのデータがサービス妨害（DoS）攻撃につながることを認めないようにすることが必要です。
- **機密性**―暗号化されたデータを傍受する盗聴者は、平文の長さ以外の平文データに関して何も特定できないようにする必要があります（意味的なセキュリティ）。
- **リプレイ攻撃からの保護**―敵対者が暗号化された正規のトラフィックを捕捉し、（場合によっては別の場所にある）ネットワークにトラフィックを再注入した場合、そのトラフィックを検出せずに宛先で受け付けないようにする必要があります。これは終端間およびリンク層の特性です。
- **DoS 抵抗**―ネットワークにパケットを注入して、ネットワークが正常に動作しないようになるまでネットワークを混雑させることは、困難にする必要があります。

30.3 SmartMesh のセキュリティ機能

- **メッセージ整合性**―メッセージの整合性は 2 つの 32 ビット・メッセージ整合性コード(MIC)を使用して実現します。1 つはリンク層(つまり、各ホップ)にあり、もう 1 つはネットワーク層/メッシュ層(つまり、メッシュ内の終端間)にあります。リンク層 MIC により、各ホップの受信側モートは、パケットがマネージャ承認済みモートによって送信されていることを確認できます。終端間 MIC は、パケットを解読して認証するために宛先で使用されます。この MIC は、送信者がパケットを送信後、パケットがどのホップでも変更されなかったことを確認します。これら 2 つの MIC は、どちらも CCM*アルゴリズムによって計算されます(以下を参照)。
- **アクセス制御**―マネージャは、ネットワークへの参加を許可されたデバイスとデバイスの参加鍵の一覧により、アクセス制御リストを維持します。デバイスは、正しい 128 ビットの参加鍵を提示せずに参加することはできません。
- **機密性**― 機密性は、ペイロードの 128 ビットの AES 暗号に基づいた CCM*ストリーム暗号で確保されます。暗号鍵とは、実行時に無作為に生成され安全に配信される共有の秘密セッション鍵です。ネットワーク外部の盗聴者がペイロードを解読することはできません。ネットワーク内部のモートが他のモートのペイロードを解読することはできません。
- **リプレイ攻撃からの保護**―ネットワークに参加するときに、各モートは持続的な参加カウンタを使用して最初のメッセージを暗号化します。リプレイ参加要求はマネージャによってすべて検出され、破棄されます。モートがネットワークに参加し、セッション鍵を取得すると、すべてのパケットは単調に増加する 32 ビットのノンス・カウンタを使用して暗号化されます。宛先では、受信したノンス・カウンタの履歴をレシーバーが追跡して、古いパケットや複製のパケットを除去します。更に、上記のリンク層 MIC はタイムスタンプ・ベースのノンスを使用して計算されます。リプレイは検出されて破棄されます。
- **DoS 抵抗**―リプレイ攻撃からの保護を実現するセキュリティ属性は、DoS 抵抗機能も備えています。

30.4 暗号化と認証

すべてのデータ・リンク層(DLL)メッセージは、CCM*と AES-128 をハードウェア内で組み合わせて使用することによって認証されます。これは、既定の鍵を使用した参加要求、および実行時ネットワーク鍵を使用したそれ以外のすべてのメッセージを使用して実行されます。DLL ノンス・カウンタは、共用時刻(ASN)に基づいてリプレイを防止します。CCM*と AES-128 を組み合わせて使用することで、ネットワーク層ヘッダを認証し、ペイロードを認証/暗号化します。参加メッセージでは、上記のように適切な対称参加鍵を使用します。モートは、リセットを通じてそのノンス・カウンタを持続します。上記のような実行時セッション鍵は、それ以外のすべての終端間セッション・トラフィックのために使用されます。各セッションは固有のノンス・データを伝送することにより、リプレイ攻撃と中間者攻撃を軽減します。

メッセージ整合性およびリプレイ攻撃からの保護は、リンク層での時間ベースのメッセージ認証によって確保されます。ネットワーク層でのセッション・ベースの認証および暗号化により、機密性、送信元の認証、およびリプレイ攻撃からの保護が確保されます。

30.4.1 キーイング・モデル

SmartMesh ネットワークでは、不揮発性(NV)フラッシュに安全に保管された 128 ビットの参加鍵が各モートにあります。モートは、この参加鍵および参加カウンタを使用して暗号化された参加要求を送信することにより、モート自体

をネットワークに対して認証します。ネットワーク・マネージャはこのメッセージを解読して、メッセージの送信元が、正しい鍵を保有しているモートであり、予想された参加カウンタを使用しているモートであることを確認します。認証が失敗した場合、参加要求は破棄されます。

デバイス認証後、マネージャは以下のセッション鍵を配布します。

- **マネージャ・ユニキャスト・セッション**—このセッションは、すべての鍵およびほとんどのネットワーク構成メッセージを安全に送信するために使用されます。モートは具体的なスケジュールに基づいて他のモートと通信します。そのスケジュールの作成およびメンテナンスに関連付けられているすべてのコマンドは、マネージャ・ユニキャスト・セッションに厳密に制限されます（つまり、マネージャだけがネットワーク資源をレイアウトすることができます）。
- **マネージャ・ブロードキャスト・セッション**—このセッションはすべてのモートに対して 1 つの鍵を使用します。また、このセッションはすべてのモートを同時に宛先とするコマンドに使用されます。このセッションは無線通信プログラミング（OTAP）に対して使用され、ネットワーク内での通知送信率を制御するために使用されます。OTAP イメージを動作状態にすることができるのは、マネージャ・ユニキャスト・セッションを使用した場合だけであることに注意してください。
- **ゲートウェイ・ユニキャスト・セッション**—このセッションはセンサー・ペイロードを暗号化して解読するために使用されます。マネージャとモートは、その暗号化機能および解読機能を実行します。
- **ゲートウェイ・ブロードキャスト・セッション**—ゲートウェイが将来ペイロードをすべてのセンサー・プロセッサにブロードキャストする必要がある場合、この鍵はそのペイロードを暗号化および解読のために使用されます。

SmartMesh IP では、管理トラフィックとデータ・トラフィックの両方に対して 1 セッションのみを使用します。

上記のセキュリティ・セッションには、それぞれ鍵があります。したがって、N モートのネットワークには、N 個のマネージャ・ユニキャスト・セッション鍵、N 個のゲートウェイ・セッション鍵、2 つのブロードキャスト・セッション鍵があります。最後の鍵は**ネットワーク MIC 鍵**です。この共有鍵は、データ・リンク層でパケットを認証するためにすべてのモートによって使用されます。言い換えると、ルーティング・モートは、パケットの送信元が有効な隣接モートであることをこの鍵を使用して確認し、パケットを解読するために必要な鍵は保有していません。

30.4.2 参加に関するマネージャのセキュリティ・ポリシー

マネージャには参加に関する 3 つのセキュリティ・ポリシーがあり、ネットワーク管理者が選択できます。

- **共通鍵の容認**—このセキュリティ・ポリシーでは、マネージャは、ネットワーク全体にわたる共有**参加鍵**を提示する**すべての**モートに対してネットワークへのアクセスを許可します。
- **ACL と共通鍵**—このセキュリティ・ポリシーでは、マネージャは、グローバルに一意の 8 バイトの MAC アドレスがアクセス制御リスト（ACL）に登録されているモートで、**かつ**ネットワーク全体にわたる共有参加鍵を提示するモートに対してのみネットワークへのアクセスを許可します。無効の MAC アドレスまたは誤った参加鍵を提示する参加要求は破棄されます。
- **ACL と固有鍵**—このモードでは、ACL 上にある各モートに固有の参加鍵があります。デバイスがネットワークへのアクセスを許可されるのは、正しい MAC アドレスおよび参加鍵を提示する場合だけです。それ以外のすべての要求は破棄されます。**これは推奨の動作モードであり、最も安全な動作モードです。**

30.4.3 鍵管理

鍵の生成と配布はネットワーク・マネージャによって実行されます。鍵の生成は NIST 規格 SP800-90 に基づきます。CTR-DRBG は熱ノイズと XOR 関数をサンプリングして、大きなランダム・シードを生成します。ネットワーク・マネージャには自動的な鍵ローテーション・ポリシーはありませんが、ネットワーク・マネージャは鍵ローテーション API を出して、すべての鍵の安全な生成およびローテーションを実行します。

30.4.4 パスワード

⚠ デフォルト・パスワードは資料内で公開されているので、パスワードの変更を強く推奨します。

SmartMesh WirelessHART マネージャと SmartMesh IP VManagers には、設定可能なパスワードが多数あります。

- Linux ログインのユーザ名とパスワード
- Linux の root パスワード
- CLI（コンソール）のユーザ名、パスワード、アクセス・レベル
- 管理ツールセットのパスワード（WirelessHART のみ）
- API のユーザ名、パスワード、アクセス・レベル

SmartMesh IP 組込みマネージャには、user（フル・アクセス）と viewer（読み取り専用アクセス）の 2 つのログイン・レベルがあります。どちらのユーザ・パスワードも変更できます。

30.4.5 CCM*に関する注意

CCM*（**C**ounter mode with a **C**BC-**M**AC）とは、カウンタ・モードと CBC-MAC の組み合わせを意味します。これでは参考にならなかったでしょうか？CBC（**C**hain **B**lock **C**ipher）とは、連鎖ブロック暗号という意味です。AES はブロック暗号で、16 バイトのデータ・ブロックを処理します。任意の長さの長いデータ（**ストリーム**）に役立てるには、**ブロック連鎖**と呼ばれる技術を使用します。1 回の AES ブロック計算の出力を次のブロックへの入力として使用するので、すべてのブロックのセキュリティは互いに「鎖でつながって」います。CCM*では、データが変更されなかったことを検証するために使用するメッセージ確認コード（**M**essage **A**uthentication **C**ode）も生成します。これはメッセージ整合性コード（MIC：Message Integrity Code）とも呼ばれます。「CCM*では」とは、認証動作、暗号化動作、あるいはその両方を実行できるという意味です。CCM は増分カウンタを**ノンス**として使用します。ノンスとは、暗号化／解読動作に対する入力として「1 回使用される数」です。ノンスを秘密にしておく必要はありませんが、送信側と受信側が両方とも現在のノンスがいくつかを（鍵と合わせて）認識し、適切にパケットを解読する必要があります。また、この数の使用は 1 回だけにする必要があります。リンク層のノンスの場合、モートは ASN（ネットワーク内で経過したスロットのカウント）を使用します。ネットワーク層では、これは増分メッセージ・カウンタです。

31 アプリケーション・ノート：TestRadio コマンドの使用

31.1 TestRadio コマンド

SmartMesh WirelessHART Mote ファームウェアは、2 つのモート間の無線リンクの質をスタンド・アロンのテストで測定する API コマンド（**testRadioTxExt** および **testRadioRx**）を内蔵しています。これらのコマンドは無線の詳細な機能を実行し、またネットワークに参加していないモートでのみ使用できます。これらのコマンドにより、インテグレータはアプリケーションのフロント・エンドを介して無線検査コマンドをプログラムで公開できるので、デバイス CLI の公開とは対照的です。

これらのコマンドは、通常は設計／製造したアンテナが期待どおりに動作することを確認する目的で使用します。このアプリケーション・ノートでは、これらのコマンドを使用して 2 つのモート間のパケット配信比（PDR）テストを行なう方法を詳しく説明します。このテストの実行対象となるのは、通常は上位の組立検査またはフィールド・デバッグ作業です。

testRadioTx コマンドおよび **testRadioRx** コマンドを使用してトランスミッタ・モートからレシーバー・モートへ 1000 パケットを送信し、受信したパケット数の統計情報を収集します。

31.2 セットアップ

以下のものが必要です。

- ネットワークには接続されていない 2 つの SmartMesh WirelessHART モートまたは SmartMesh IP モート
- 使用可能な 2 つの USB ポートを備え、SmartMesh SDK がインストールされたコンピュータ
- 2 本の USB ケーブル

起動する前に、コンピュータに接続されている 2 つのモートの API ポートに対応するシリアル・ポート番号を書き留めてください。

31.3 実験の実施

- APIExplorer アプリケーション（SmartMesh SDK の一部）の 2 つのインスタンスを起動します。
- アプリケーションを両方とも 2 つのモートの API ポートに接続します。
- トランスミッタ・モートに接続されている APIExplorer で、以下の操作を行います。
  - **testRadioTxExt** コマンドを選択し、以下に示すフィールドにデータを取り込みます。
    - **testType**：パケット - これにより、モートは IEEE802.15.4 準拠のパケットを送信するよう指示されます。
    - **chanMask**：8000 - これは チャンネル 15 つまり 2.480GHz でパケットを送信することを示しています（`0x8000` はビットマップ `b1000000000000000` に対応します。つまり、ビット 15 だけが設定されます）。

- **repeatCnt**:1 - パケット・シーケンスを繰り返す回数。
- **txPower**:8 - モートは+8dBm の伝導電力でパケットを送信します。
- **seqSize**:1000 - 各シーケンスでのパケットの数
- **sequenceDef**:pkLen = 125、delay = 20000。パケット間の間隔は 20 ミリ秒なので、1000 パケットを送信するのに 20 秒かかります。モートはパケットの末尾に 2 バイトの CRC を追加するので、結果として最大長の IEEE802.15.4 パケットになります。

- 「**send**」ボタンはまだクリックしないでください。

- レシーバー・モートに接続されている APIExplorer で、以下の操作を行います。
  - **testRadioRx** コマンドを選択し、以下に示すフィールドにデータを取り込みます。
    - **channel**:2.480GHz。これはチャンネル 15(トランスミッタが送信しているチャンネル)に対応します。
    - **time**:30.これにより、モートは 30 秒間パケットを受信するよう指示されます。トランスミッタは 20 秒間送信するので、これによって小容量のバッファが得られます。
- 以下のようにしてテストを開始します。
  - レシーバー側の「**send**」ボタンをクリックします。モートは 30 秒間の受信を開始します。
  - トランスミッタ側の「**send**」ボタンをクリックします。モートは、全 20 秒間にわたる 1000 パケットの送信を開始します。
- 30 秒間待機し、以下の手順で結果を収集します。
  - レシーバー・モートに接続されている APIExplorer で、**getParameter** コマンドと **testRadioRxStats** サブコマンドを入力します。「**send**」を押します。
  - 応答フィールドには以下のフィールドが含まれます。
    - 「**rxOk**」は、正常に受信したパケットの数です。
    - 「**rxFail**」は、受信したが CRC が失敗したパケットの数です。これは、パケットを受信したが送信中に破壊されたことを示しています。

31.4 結果の解釈

干渉、マルチパス・フェージング、不適当なアンテナ接続、送信側と受信側との距離が離れすぎている場合など、いくつかの物理的現象はパケットを正確に受信しない原因になる場合があります。これらの場合にはパケットを低 SNR で受信できるので、結局パケットが破壊(して、**rxFail** カウンタで表示)するか、まったく受信しなくなる可能性があります。

したがって、以下のことを確認する必要があります。

- 正しく受信されたパケットの数は？これは **rxOk** カウンタで確認します。
- 受信したが破壊しているパケットの数は？これは **rxFail** カウンタで確認します。
- 失われたパケットの数は？これは、送信したパケットの数と、**rxOk** カウンタと **rxFail** カウンタの数の合計との差です。

32 アプリケーション・ノート:ピーク時の平均電流を制限するための最良実施例

LTC5800 は低消費電力設計を目的としています。ただし、通常のモートは 50μA 未満の動作電流が標準的ですが、(デバイスの起動時など)間隔が短い場合は電流がはるかに大きい可能性があります。このことは、電流が制限された電源で動作する設計に対して課題を突きつけています。LTC5800 は、デバイス自体を 360μA の平均電流に制限することを目標として設計されています。この平均電流は、これらのピーク電流間隔時に 7 秒間にわたって平均化された値です。ただし、この電流目標値を満たすには、ある特定のシステムレベルの考察を重んじる必要があります。この電流レベルは、従来の SmartMesh WirelessHART 製品(DN2510)での低電流起動モードに対応しており、このモードは 4～20mA の電流ループでのモートの動作をサポートすることを目的としていました。

特に強調していない場合、各項目はすべての SmartMesh ファミリに当てはまります。

- **リセット** – 起動時には約 800μA を消費し、800 ミリ秒かかります。起動後にデバイスを繰り返しリセットすると、平均電流が目標値より高くなることがあります。OEM プロセッサを 2 秒より短い間隔では再起動しないというのが弊社のガイドラインです。
- **ジョイン・デューティ・サイクル** – 平均電流がこの目標値を満たすように、ジョイン・デューティ・サイクルは 5%以下に設定します。
- **UART 動作(SmartMesh WirelessHART)**– UART トラフィックは 9600bps では電流に多少影響があります。ある 1 つの API での処理を 115kbps で連続ループすると、平均電流が約 75μA 上昇するので回避する必要がありますが、これは通常の使用事例ではありません。デバイスの通常の使用では、ポーリングではなく通知が使用されます。時刻ピンのポーリングは 2 秒より短い間隔で行わないようにしてください。
- **持続パラメータ** – このシステムでは、**setNVParameter** API を使用してフラッシュ・ベースの不揮発性パラメータを変更できます。ただし、書込みが連続すると平均電流の目標値を超えることがあります。弊社のガイドラインは以下のとおりです。
  - 起動通知の受信後、2 秒待ってから NV 書き込みを実行する
  - NV パラメータの連続的な書き込みは、最短でも 2 秒間隔で行う必要がある
  - NV 書き込みを起動通知後に行う場合は、参加要求を出す前に 2 秒待つ
- **OTAP** – OTAP では、(着信する OTAP ファイルを保管するために)ファイル・システムを消去し、(稼働中のファームウェアを置き換えるために)メインのフラッシュ領域を消去する必要があります。現在のリリース(SmartMesh WirelessHART 1.0.2、SmartMesh IP 1.2.0)では、この動作は不可分であり、電流の目標値を超えるので、電流が制限されたデバイスでは、このバージョンのコードを使用した場合、OTAP を回避する必要があります。OTAP による消去は、後続のリリースでは動作が周期化され、電流の目標値を満たします。OTAP は頻度の低い動作であり、寿命に達するまでにアップグレードされないシステムが多数を占めます。
- **通電バックボーン(SmartMesh IP 組込みマネージャ、SmartMesh WirelessHART のマネージャ 4.1 以上)** – DN2510 の低電流起動モードは、このネットワーク・モードでの電力の目標値を満たしません。LTC5800 モートは、IP ネットワークと WirelessHART ネットワークの両方のバックボーン上で非ルーティング・リーフとして動作できます。
- **極度の輻輳** – マネージャはモートのリンクを制限して、すべてのリンクが使用された場合でも電流の目標値を超えないようにします。実際に使用されるのはこれらのリンクの一部だけなので、現在のリリースでは、温度変動による無線の再調整など、低頻度のハウスキーピング動作は考慮していません。非常にまれな事例

では、デバイスがその無線を再調整する必要がある場合、極度に輻輳したモートが目標値を超えることがあります。これによってリセットが発生した場合でも、モートは起動時に再調整して正常に参加します。

- **スクラッチ・パッドの使用（SmartMesh WirelessHART モート 1.1 以上）** – バージョン 1.1.x 以降のモート・ソフトウェアでは、顧客の使用を目的としてスクラッチ・パッドを使用できます。ガイドラインは持続パラメータの場合と同じです。平均電力の目標値を確実に満たすため、スクラッチ・パッドの書き込みは 2 秒以上の間隔を空ける必要があります。

33 アプリケーション・ノート：パイロット・ネットワーク評価の方法

33.1 まとめ

パイロット・ネットワークの配置、テスト、分析を行い、最終的な配置先での典型的な環境で十分に機能するかどうかを確かめたいという要求がよくあります。モート／センサーの場所、トポロジ、報告率などに関してパイロットの配置が実際の配置と一致する割合が近づくほど、実際のネットワークがどのように機能するかの感触をつかみやすくなります。

SmartMesh SDK は、固有の環境でネットワークを評価するシンプルなツールを提供します。ここでは、検査用の特有な環境でモートを配置するのに役立つ段階的なアプローチの概要を説明します。配置後、SDK に付属のユーティリティを使用して、ネットワーク内の様々なモートを構成し、所望の率でデータを報告するよう設定できるので、動作を報告する実際のセンサーを実際の配置でエミュレートできます。本書では、配置後のネットワークから一定期間にわたって統計情報を収集して分析し、信頼性、データ・レート、帯域幅などの重要なパラメータに関してパイロット・ネットワークの性能を評価する方法について説明します。

33.2 SmartMesh スタータ・キットと SDK

SmartMesh スタータ・キットには、標準で 1 つのマネージャと 5 つのモートが付属しています。必要に応じて追加のモートを購入して、ネットワーク・サイズを大きくすることができます。アナログ・デバイセズの FAE との事前の協議に基づいて、用途に応じて適切な製品ファミリ（SmartMesh IP または SmartMesh WirelessHART）を決定する必要があります。SmartMesh IP を選択する場合、このアプリケーション・ノートで説明するのは組込みマネージャでのスタータ・キットの使用法であり、VManager ではありません。すべての資料、ソフトウェア・ユーティリティ、およびツールは http://www.linear.com/starterkits でダウンロードできます。

SmartMesh IP Easy Start Guide または SmartMesh WirelessHART Easy Start Guide は、スタータ・キット・ハードウェアの設置と SDK ソフトウェアの基本機能のインストールについて説明しています。このアプリケーション・ノートでは、これ以降、PC に適切な SDK が適切にインストールされており、マネージャで CLI（コマンド行インターフェース）を操作できることを前提とします。

> ⚠ 設置やインストールの詳細については、SmartMesh IP Tools Guide または SmartMesh WirelessHART Tools Guide を参照してください。

33.3 評価方法

パイロット・ネットワークは、基本的に以下の 5 段階で評価します。

- ステップ 1 - SDK の PC へのインストール

- ステップ 2 - SDK のモートおよびマネージャのパイロット・ネットワークの配置
- ステップ 3 - 特定のデータ・レートに対応したモートの構成
- ステップ 4 - データと統計情報の収集
- ステップ 5 - データの分析

33.4 ステップ 1：SDK の PC へのインストール

1. http://www.linear.com/starterkits から SDK をダウンロードします。目的の製品ファミリ（SmartMesh IP または SmartMesh WirelessHART）に基づいて、該当する資料をダウンロードします。
2. Easy Start Guide を確認し、SDK を PC にインストールします。
3. PC をマネージャに接続して CLI を実行できることを確認します。
4. SmartMesh IP の場合は、シリアル・マルチプレクサが設置されていて、正常に動作していることを確認します。詳細なインストール手順については、SmartMesh IP Tools Guide を参照してください。

33.5 ステップ 2：SDK のモートおよびマネージャのパイロット・ネットワークの配置

- アプリケーション・ノート配置の計画を確認します。重要な配置ガイドラインは、各モートを他の 3 つの隣接モート／モートの範囲内に置くことです。
- 物理的環境に基づいて範囲を決めます。50 メートルの範囲から始めて、結果に応じて後で調整してもかまいません。

⚠ ネットワークの解析では、設置する各モートの場所、または少なくとも隣接モートまでの距離を文書化できれば非常に役立ちます。しばらくの間ネットワークを稼働したら、RSSI と対照してグラフを作成するためには、この距離データが非常に重要になります。

- 上記の 3 つの隣接モートのガイドラインと対象にする領域に基づいて、パイロット・ネットワークの配置に必要なモートの数を決めます。正常な配置のために必要な数のモートを保有していることを確認してください。
- マネージャを最初に配置します。PC を介してマネージャにアクセスできることを確認します。マネージャに電源を入れ、CLI コマンドを実行できることを確認します。
- モートのバッテリが新しいことを確認します。モートは新しいバッテリを装着して出荷されていますが、SDK のインストール時やテスト時に消耗した可能性があります。この状況が起こる可能性があるのは、マネージャ／ネットワークが使用できない状態でモートが放置された場合です。この場合、モートは、容量を使い果たす可能性がある時間の 25%で、レシーバーを使用してネットワークを探索し続けます。
- 特定のパラメータをモニタするために、センサーを配置する可能性が高い場所にモートを配置することから始めます。電源スイッチをオンにします。SmartMesh IP モートを有効化した場合は、Status_0 LED が点滅し始めます。
- モートの MAC アドレスと適切な配置先を必要に応じてマップまたは表に書き留めます。こうすると、ステップ 3 でデータ・レートに合わせてモートを構成するときに役立ちます。

⚠ モートを設置する前に、モートが**マスタ・モード**で構成されていることを確認してください。これはモートのデフォルト設定ですが、SDK のテスト中に構成を変更した可能性があります。

- 必要に応じて、計画のアプリケーション・ノートで概要を説明した方法に従って、中継器として動作する追加のモートを設置してください。
- SmartMesh IP の場合、すべてのモートが配置されると 2 つのステータス LED が点灯し、各モートが現在マネージャに接続されていることを示します。
- 今度はマネージャに戻り、ネットワークが形成されていることを確認します。これはマネージャの CLI 接続を介して実行できます。コマンド `sm`（「show motes」の省略形）を使用して、ネットワーク内のモートの一覧を表示します。

⚠ いったん配置すると、ネットワークのサイズによりますが、ネットワークを形成するのに数分かかる場合があります。モートが参加するように見えない場合は、モートを再起動してみてください。CLI コマンド `trace motest on` を使用して、モートの状態変更をトレースすることもできます。これにより、モートが参加処理を段階的に進めるのに応じて通知が書き込まれます。

下の図 1 には、マネージャに参加した 5 つのモートを、`sm` によって表示された状態で示します。モート ID 1 は AP（アクセス・ポイント）モートです。ネットワーク内のすべてのモートがこの一覧に表示されることを確認してください。

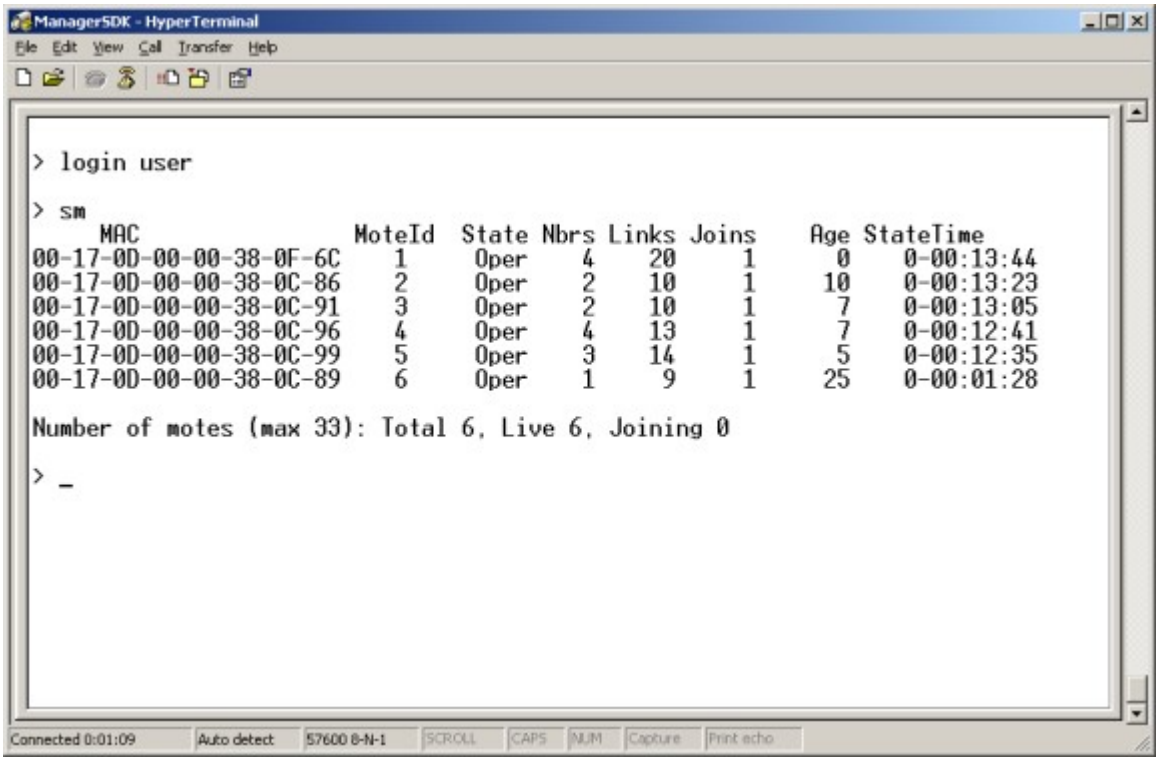


図 1 – マネージャに接続されているモートを示す CLI コマンド `sm`

33.6 ステップ 3：特定のデータ・レートに対応したモートの構成

ネットワークは稼働状態になっているので、データを生成して送信するようモートを構成して、あたかもモートがセンサーからデータを収集しているかのようにする必要があります。ネットワークを介して上り方向（モートからマネージャ）に送るデータの量、レート、およびサイズを指定します。SDK で利用可能な PkGen ユーティリティを使用してこれを遂行します。

パケット生成ユーティリティ PkGen では、ネットワーク内の各モートの構成をマネージャ自体から行うことが可能です。これは、個々のモートの場所に物理的に移動してモートのレートを設定する必要がないことを意味します。また、これらの構成を必要に応じて変更するには、変更が必要なモートの新しいフィールドに入力するだけで済みます。SmartMesh ネットワークは異機種ネットワークとして動作できるので、各モートはモート固有のデータ・レートに合わせて構成できます。データ・レートが低速/中速/高速のモートを組み合わせて、低速/中速/高速センサーの現実世界のシナリオをエミュレートすることができます。

PkGen ユーティリティを使用してモートを構成するには、以下の手順に従ってください。

> ⚠ このセクションでは、serialMux を動作状態にしておく必要があります。

- SmartMesh SDK の`/win/`ディレクトリに移動し、`/win/PkGen.exe` という Windows 実行可能プログラムをダブルクリックすることにより、PkGen を起動します。
- SmartMesh IP または SmartMesh WirelessHART を選択するよう指示されます。該当するファミリを選択して、「**load**」をクリックします。
- IP デバイスに接続している場合は、指示されたときに serialMux を介して「connect」をクリックすれば済みます。これが動作しない場合は、serialMux がステップ 1 で正しく設置されていることを確認してください。
- WirelessHart デバイスに接続している場合は、指示されたときにマネージャの IP アドレスを入力します。WirelessHART マネージャのデフォルトの静的 IP アドレスは 198.162.99.101 です。このアドレスは任意のアドレスに変更できます。詳細については、SmartMesh WirelessHART User's Guide を参照してください。
- 接続すると、入力したフィールドが緑に変わり、ネットワーク内のモートの一覧がモート一覧のセクションに取り込まれます。ネットワーク内の各モートにはテーブルに一連のフィールドがあり、ここには各モートを構成するために必要に応じて情報を入力します。
- **mac** 列は、ネットワークに接続されているモートの MAC アドレスを示しますが、これは構成の対象です。これを使用して現場でのモートの位置を割り出し、その後、具体的な構成を決定する場合があります。
- 次の列（**num. pkgen**）は、PkGen の起動後任意の時点でそのモートからマネージャに送信されたパケットのカウンタです。
- 次の列（**pk./sec**）は、モートがマネージャにパケットを送信しているときのパケット/秒単位の平均レートを示す別のカウンタです。ここに値が表示されるまでしばらく時間がかかる場合があります。
- 前の 2 列は「**clear**」ボタンを押すとリセットできます。この操作を毎回行ってから、送信パラメータを新たに設定することになるでしょう。
- 最後の列には 3 つのフィールドと「**set**」ボタンがあります。最初のフィールドは、モートが送信するパケット数の合計用です。配置の期間とモートがデータを送信するレートに応じて、適切な数値をここに入力してください。少なくとも 1 つのパケットを送信する必要がありますが、最大値はありません。2 番目のフィールドはパ

ケットの間隔で、単位はミリ秒です（**例えば**、1000 = 1 パケット/ 1000 ミリ秒つまり 1 パケット/秒）。最後のフィールドは、毎回送信するパケットのサイズです。（最大 60 バイトで 20 バイトのヘッダあり）。

⚠ 空白のセルに「-」を入力することはしないでください。空白のままにしておいてください。

- 最後の列の「**set**」ボタンをクリックして、モートによるパケットの送信を開始します。（必要な場合は、「**clear pkgen**」ボタンを使用すればこれらのフィールドをリセットできます。）

⚠ 「**set**」ボタンを押してから次に押すまで時間をおいてください。「**set**」を 2 回急に押すと、セルが誤って緑に変わり、セルが更新されたように表示されますが、パケットはネットワークに送信されません。

- ネットワーク内のモートごとに上記の 2 手順を繰り返します。

⚠ マネージャがパケットの受信に利用できる全帯域幅には上限があります。この値はファミリに依存しますが、約 25 パケット/秒です。この限度は超えないようにしてください。言い換えると、すべてのモートからマネージャに移動する 1 秒当たりのデータ・パケットの合計がこの限度を超えないようにする必要があります。個々のモートのデータ・レートは、このようにネットワーク内のモートの総数とモートの個別データ・レートに依存します。例えば、5 つのモートを備えたネットワークでは、すべてのモート上で同時に 5 パケット/秒を超えないようにする必要があります。

- スクリプトを閉じても、モートがパケットの送信を停止するという意味にはなりません。モートはすべてのパケットを送信するまで送信し続けます。モートによるデータの送信を停止する場合は、モートをリセットしてください。

下記の図 2 は、PkGen スクリーン・ショットの例を示します。例えば、最大サイズのパケットを 1 週間 1 パケット/秒のレートで送信する場合は、「**num/rate/size**」フィールドに次のように入力します。**604800 1000 60**

パラメータをモートごとに好みの値に設定すると、「**num. pkgen**」カウンタが計数を開始するのを確認できます。「**pk./sec**」列には、最も近い 0.1 パケット/秒単位の値に丸められたレートが表示されます。レートが低すぎるか、送信されるパケット数が（平均を計算するには）不十分な場合は、0.0 というレートが表示されます。

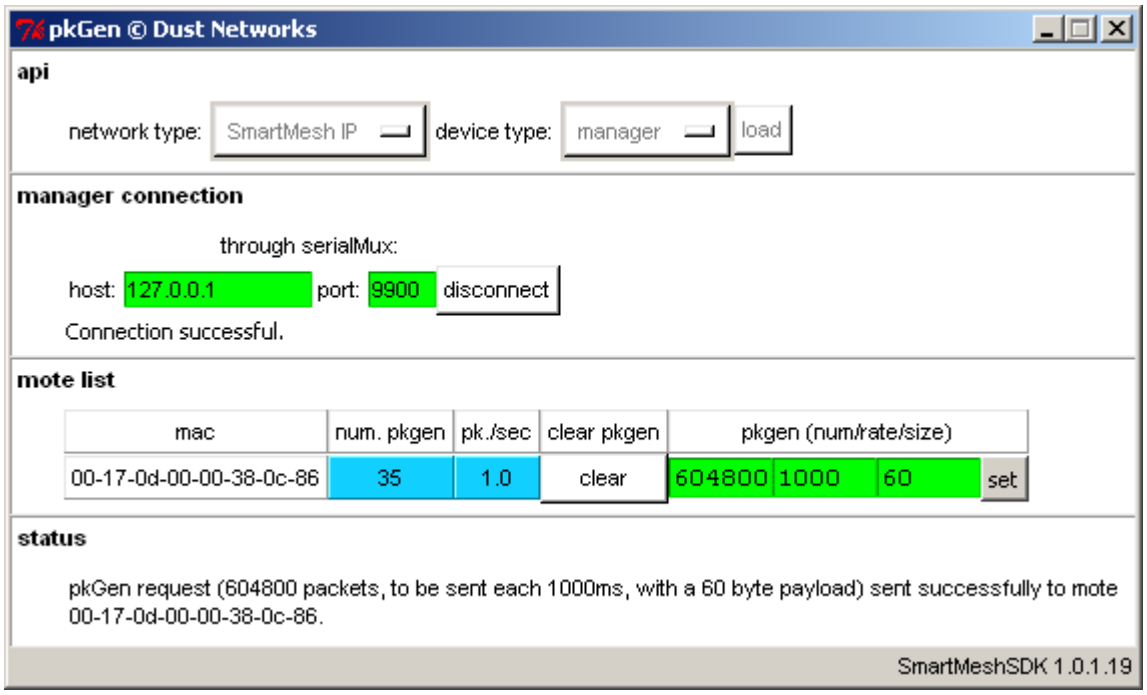


図 2 - モートの構成を示す PkGen スクリーン・ショット

33.7 ステップ 4：統計情報の収集

ネットワークは上り方向にデータを送信しているので、次の段階は、現在の構成でのネットワークの性能を分析するために使用できるネットワーク統計情報を収集することです。ネットワーク内の各モートは、ネットワーク、パケット成功率、隣接モートなどに関する情報が含まれているパケットを定期的に送信します。これらのパケットは**健全性レポート**と呼ばれ、標準で 15 分おきに送信されます。SmartMesh ネットワークは、データ・アクセスと同様に、すべてのネットワーク統計情報にアクセスできる API に構築されています。

SmartMesh WirelessHART と SmartMesh IP のどちらを使用しているかに応じて、2 つのうちいずれかの方法で統計のスナップショットをとります。

ステップ 4A：IP ネットワークでの統計情報の収集

SmartMesh IP マネージャは、ネットワーク信頼性、経路安定性、およびモート動作の最小限の統計情報一式を維持します。ネットワーク健全性の全体像をつかむには、すべての健全性レポート通知を記録する必要があります。このトピックの詳細については、アプリケーション・ノート：SmartMesh IP ネットワークの健全性のモニタで説明します。

ステップ 4B：WirelessHART ネットワーク・スナップショットによる統計情報の収集

SmartMesh WirelessHART マネージャは、すべてのマネージャ・ログを収集するユーティリティを内蔵しています。このユーティリティを呼び出すには、マネージャにログインし、Linux プロンプトでコマンドを使用します。

1. CLI に接続する方法である dust ログインを使用してマネージャにログインします。
2. nwConsole を使用する代わりに、次のコマンドを使用します。

```
/usr/bin/create-network-snapshot
```

これにより、現在のネットワーク統計情報のスナップショットを取って、それを `/tmp/snapshot/snapshot.tar.gz` に保管できます。

1. このスナップショットにアクセスするには、WinSCP または別の FTP クライアントを使用してマネージャに接続します。pkgen およびポート 22 との接続に使用したのと同じ IP アドレスに接続します。`/tmp/snapshot` に移動して、`snapshot.tar.gz` を自分のマシンに転送します。

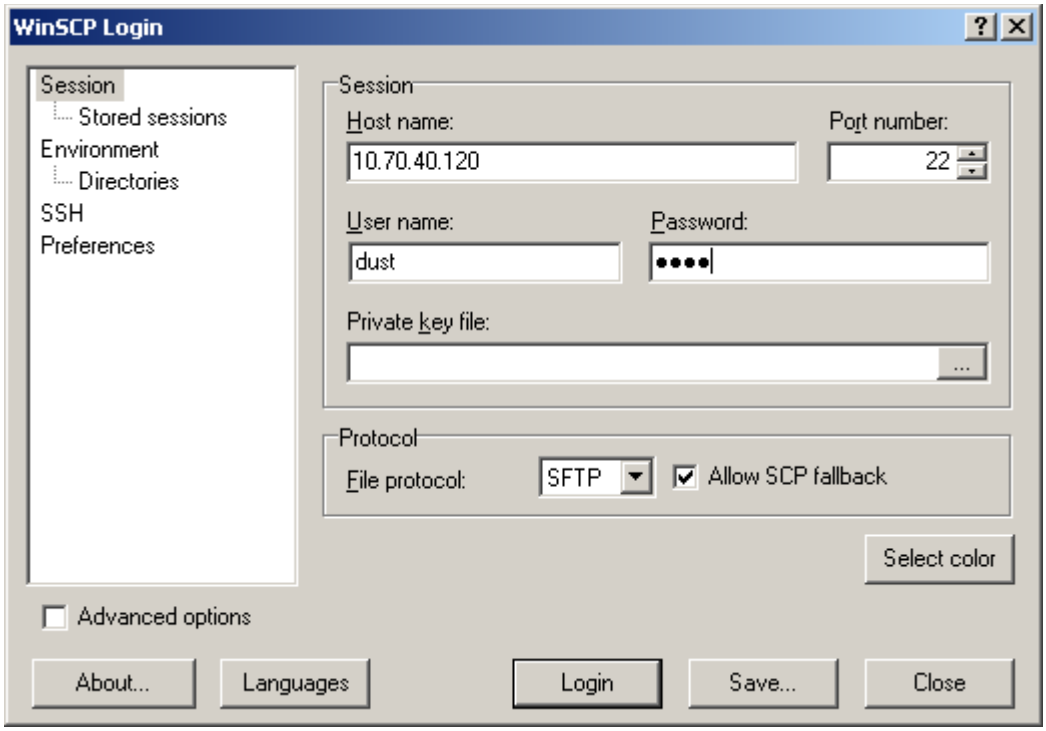


図 3 - マネージャからのネットワーク・スナップショットの転送

⚠ 新しいスナップショットを取ると古いスナップショットは上書きされるので、必ずファイルをモートから転送してから新しいスナップショットを取るようにしてください。

33.8 ステップ 5:データの分析

SmartMesh IP でのスナップショット・ログの収集を効率化し、またスナップショット機能によって生成されたログを分析する実例ツールは、まだ開発中ですが、供給可能になると SmartMesh SDK の一部になります。

スナップショット・ログを検査する 1 つの目標は、経路 RSSI に関するデータを取得することです。これにより、このデータを距離と比較して、より優れたネットワークを構築できます。より信号強度が高い経路を探すことにより、最適なモート間隔をうまく判断することができます。

33.8.1 WirelessHart スナップショット・ログ

このログは、マネージャから抽出した.tar ファイルにあります。このファイルを解凍すると、求めているデータは `nwconsoleOut.txt` ファイルにあることが分ります。このログは、その名前のとおり、実際にはマネージャのコンソールの内部照会の出力にすぎません。スナップショットの時点でのネットワークの状態に関する情報を集める様々なコマンドを出します。`show ver` から始まる最初のコマンドを実行すると、おおむねマネージャのバージョン、状態、および設定に関する基本的な情報が得られます。

`sm -a` を実行すると、ネットワーク内のモートの一覧が出力されます。

以下に示すのは、その出力の一部です。

```
< sm -a
Current time: 07/24/12 09:15:03 ASN: 32492664
Elapsed time: 3 days, 18:15:30
MAC MoteId Age Jn UpTime Fr Nbrs Links State
00-17-0D-00-00-1B-1B-CD ap 1 1 3-18:15:23 6 10 106 Oper
00-17-0D-00-00-38-0C-86 2 16 2 19:02:46 2 2 11 Oper
00-17-0D-00-00-38-0C-89 3 9 2 1-00:03:00 2 7 17 Oper
```

以下に示す `get paths` コマンドを実行すると、各モート対と、その経路上のリンクの数および方向、ならびに経路品質が一覧表示されます。リンクが存在しない場合は unused(未使用)と表示され、経路品質はデフォルトの 75 になります。

```
< get paths
pathId: 65538
moteAMac: 00-17-0D-00-00-1B-1B-CD
moteBMac: 00-17-0D-00-00-38-0C-86
numLinks: 5
pathDirection: downstream
pathQuality: 90.07
pathId: 131077
moteAMac: 00-17-0D-00-00-38-0C-86
moteBMac: 00-17-0D-00-00-38-0C-96
numLinks: 0
pathDirection: unused
pathQuality: 75.00
```

この後は、便利だが該当する情報が少ないコマンドが続きますが、その一方で、必要な情報の大半を占める `show mote -a` コマンドがあります。これらのコマンドは、各モートの隣接モートまでのすべての経路と、これらの経路のそれぞれの RSSI および経路品質も表示します。

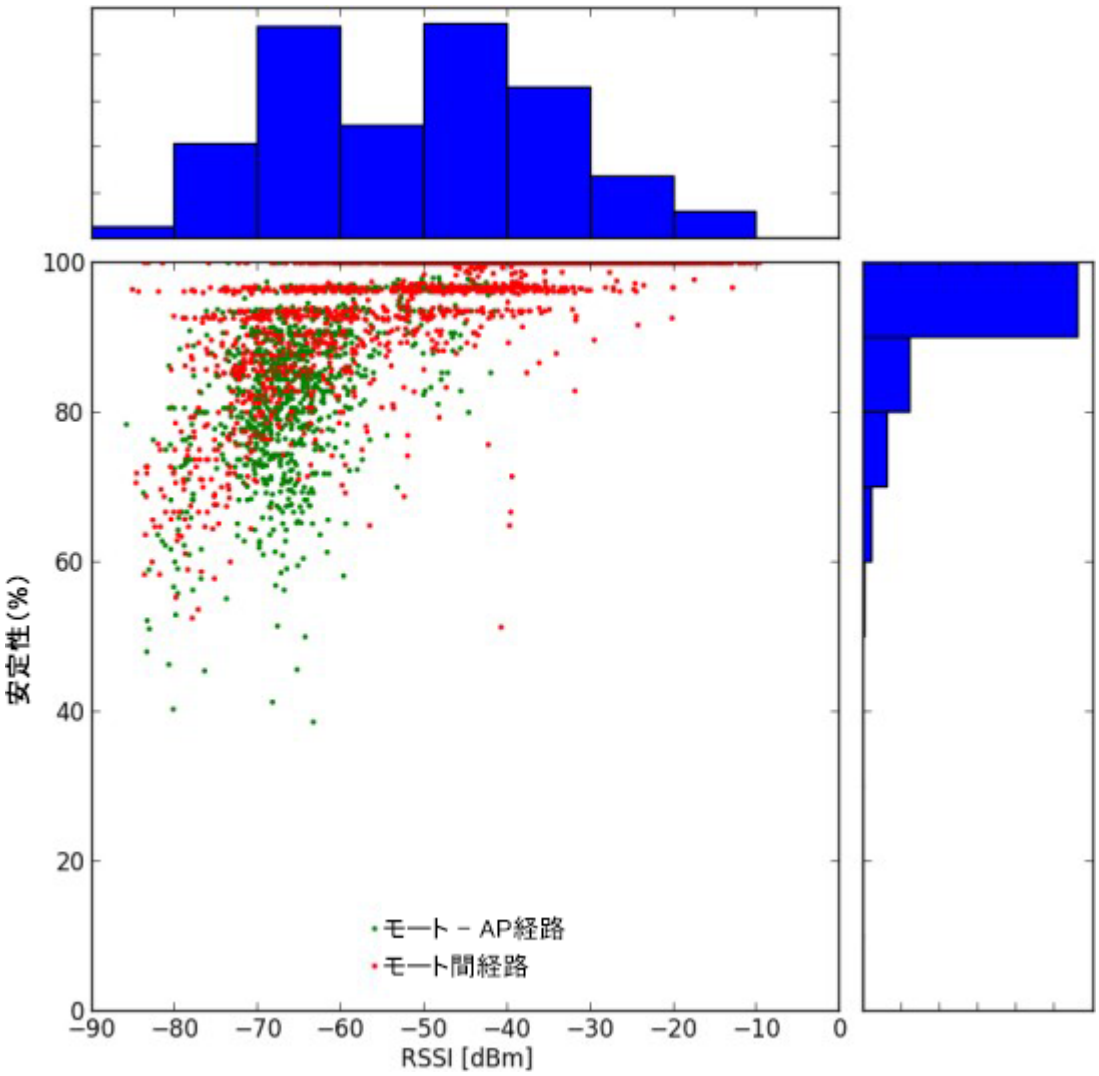
```
< show mote 1 -a
00-17-0D-00-00-1B-1B-CD 1 Oper SW: 3.0.2-0 HW: 37
Location is not supported
Number free TS: 757
Upstream hops: 0, latency: 0.000, TTL: 127
SourceRoute: Dist(Des): 0.0(10) Prim: 1 Sec:
Power Source: line
Advertisement Period: 20.000
Bandwidth:
Summary for AP: 2.5342
Neighbors: 10. Links: 106. Norm/bitmap 21/85 (max norm: 63)
Links per second: 19.824219 (unlimited)
Frame: 0. Neighbors: 10. Parents: 0. Links rx:87, tx:1.
Broadcast links
0. 1. 0: rtdb
0.993.10: rjb
<- #2 Links 3/3/3 RSSI: -52 Q: 0.90
<- #3 Links 4/4/4 RSSI: -29 Q: 0.94
<- #4 Links 8/8/8 RSSI: -50 Q: 0.83
<- #5 Links 5/5/5 RSSI: -51 Q: 0.86
<- #6 Links 3/3/3 RSSI: -48 Q: 0.90
<- #7 Links 33/33/33 RSSI: -41 Q: 0.86
<- #8 Links 16/16/16 RSSI: -43 Q: 0.90
<- #9 Links 6/6/6 RSSI: -45 Q: 0.87
<- #10 Links 4/4/4 RSSI: -41 Q: 0.90
<- #11 Links 3/3/3 RSSI: -41 Q: 0.95
```

この部分以降、ログには多くの統計表が取り込まれます。その内容は、モート、経路、およびネットワークの日々の統計の他に、それらの寿命統計が含まれます。

33.8.2 ログ・ファイルの解釈と分析

目標は、距離対 RSSI（経路安定性）を示す滝型曲線を生成することです。この曲線の屈曲部に着目すると、配置環境内でどの範囲が適切かについていい考えが浮かびます。この範囲は、元の配置で使用されたデフォルトの 50m に相対するものとして、将来の配置に使用してもかまいません。

グラフの例を以下に示します。



これらのグラフは以下の方法によって生成できます。

- RSSI（経路品質）の値を経路ごとにログ・ファイルから抽出する
- 各モート対の間の距離を計算する
- 各 RSSI（経路品質）の値と、対応する距離とを対比させて作図する

33.8.3 隣接モートの安定性分析

モートが備える良質な隣接モートの数（強い経路安定性）は、回復力のあるネットワークを形成するのに非常に重要です。このデータをログ・ファイルから収集すると、ネットワーク・メッシュ内にある弱いリンクを見つけるのに役立つ場合があります。3 つ以上の良質な隣接モートを持つモートはネットワークに正常に参加する確率が高いので、このフィードバックはネットワークの配置担当者にとって特に役立ちます。良質な隣接モートの数が 3 未満の場合は、より多くの隣接モートを近くに配置する必要性が高くなります。

その他のネットワーク健全性および性能を獲得する方法については、該当するファミリのアプリケーション・ノート：ネットワーク性能とデバイス性能の評価方法に説明されている場合があります。

34 アプリケーション・ノート：パケット ID の概要と必要な理由

34.1 適用範囲

本書では、WirelessHART と SmartMesh IP の両モート製品のモート API ヘッダにあるフィールド・フラグでのパケット ID（ビット 1）のユーティリティについて説明します。本書の最初のセクションでは、パケット ID について概説します。2 番目のセクションでは、モートとセンサー間のプロセッサ・インターフェースで 2 つのパケット ID がどのように使用されているか、それぞれがどのように独立して切り替わり、追跡されるかについて説明します。3 番目のセクションでは、パケット ID および関連する同期ビットをコードでどのように使用するかについていくつかの有益なヒントを示します。

34.2 パケット ID とは？

電動式のバケツ給水機があり、通信インターフェースがあって、このインターフェースを介して装置に以下の指示を出すと想定します。

1. 右へ移動しなさい
2. そこにバケツがあることを確認しなさい
3. バケツ 1 杯分の水を注ぎなさい
4. これを繰り返しなさい

これをバケツ給水機に指示するマイクロコントローラ・アプリケーションは作成可能であり、通常は正常に機能します。このインターフェースを通信エラーに対してより堅牢にするために、**呼び出し応答**プロトコルを実装すると想定します。クライアント（ここではマイクロコントローラ）からの各コマンドに対して、サーバ（ここではバケツ給水機）からの応答を生成します。トランザクションは以下のように見え始めます。

1. マイクロコントローラ：右へ移動しなさい
2. バケツ給水機：右へ移動しました
3. マイクロコントローラ：そこにバケツがあることを確認しなさい
4. バケツ給水機：バケツがあります
5. マイクロコントローラ：バケツ 1 杯分の水を注ぎなさい
6. バケツ給水機：バケツ 1 杯分の水を注ぎました
7. これを繰り返しなさい

クライアントはそのメッセージが正常に伝達されたことを確認するので、より堅牢なインターフェースが得られました。ただし、メッセージが届かない場合は、別の問題が予測されます。マイクロコントローラが「右へ移動しなさい」というコマンドを送信し、応答を受信しなかった場合、これはバケツ給水機がコマンドを受け取っていないか、またはマイクロコントローラが応答を受け取っていないことが原因の可能性があります。前者の場合は、クライアントがコマンドを繰り返す必要があるのに対して、後者の場合はサーバの混乱を避けるため、次のコマンドに移る必要があります。必要なのは区別を可能にする識別子です。コマンドと一緒にカウンタを送信することによってこれを実現できます。

1. クライアント：右へ移動しなさい（ID = 0）
2. サーバ：右へ移動しました（ID = 0）
3. クライアント：そこにバケツがあることを確認しなさい（ID = 1）- コマンドが届いていない
4. 応答を受信していないので、クライアントは最後のコマンドを繰り返します
5. クライアント：そこにバケツがあることを確認しなさい（ID = 1）
6. サーバ：バケツがあります（ID = 1）
7. クライアント：バケツ 1 杯分の水を注ぎなさい（ID = 2）
8. サーバ：バケツ 1 杯分の水を注ぎました（ID = 2）- 応答が届いていない
9. 応答を受信していないので、クライアントは最後のコマンドを繰り返します
10. クライアント：バケツ 1 杯分の水を注ぎなさい（ID = 2）
11. サーバ：給水機はコマンド（ID = 2）に応答してバケツを既に水で満たしたので、コマンドを破棄して「バケツ 1 杯分の水を注ぎました（ID = 2）」と応答したことを認識しました

コマンドに識別子があったので、コマンドの受信側は再試行と新しいコマンドとを区別できます。バケツ給水機はコマンドで別の ID を受信しなかったので、再度バケツに水を注ぐ必要がないことを認識していました。現在のコマンドにアクノリッジが返されるまでは次のコマンドに進まない**信頼できる**インターフェースでは、コマンドと後続のコマンドを区別することだけが必要なので、1 ビット・カウンタを ID 用に使用することができます。コマンド 1 はコマンド 0 の後に発行され、コマンド 0 はコマンド 1 の後に発行されます。これが 1 ビットのパケット ID フィールドが機能する仕組みです。パケット ID ビットを 0 と 1 の間で切り替えることによって、受信側は再試行と新しいコマンドとを区別できます。ほとんどのモート・コマンドでは、バケツに 2 杯分の水を注ぐことや、右に 1 歩ではなく 2 歩動くような悪い結果になることはありませんが、これは依然としてインターフェースの信頼性に対する大きな改良点です。

34.3 2 つのパケット ID の存在

モートとセンサー・プロセッサを内蔵しているデバイスでは、2 つの独立した通信ストリームがあります。1 つはモートがサーバでセンサー・プロセッサがクライアントである場合で、もう 1 つはセンサー・プロセッサがサーバでモートがクライアントである場合です。各ストリームにはそれ固有のパケット ID があります。センサー・プロセッサは、モートにコマンドを送信するたびに、そのパケット ID を切り替える必要があります。モートはコマンドを受信するたびにそのパケット ID を検査し、前に受信したパケット ID と比較します。パケット ID が格納値と異なる場合はコマンドが処理され、このパケット ID が格納されます。パケット ID が重複している場合、コマンドは処理されません。また、モートは、前のコマンドで送信したのと同じ応答（キャッシュに格納されている応答）で応答します。

モートはセンサー・プロセッサにコマンド（通知）をいつでも送信できるので、その通信には第 2 のパケット ID が存在します。モートはコマンドを送信するたびに、そのパケット ID を切り替えます。センサー・プロセッサはパケット ID を検査し、前に受信したパケット ID と比較する必要があります。パケット ID が格納値と異なる場合は通知が処理され、このパケット ID が格納されます。パケット ID が重複している場合、通知は破棄され、マイクロコントローラはキャッシュに格納されている前の応答を送信します。

34.4 同期ビット

センサー・プロセッサ・アプリケーションは、リセットするか、何らかの理由がある場合、使用または想定するパケット ID がわからなくなります。同期ビットを使用すると、クライアントがパケット ID をリセットして既知の状態にすることができます。センサー・プロセッサは、モートに送信する最初のパケットに同期ビットを設定します。これにより、このパケットはパケット ID にかかわらず新しいパケットであるとモートに伝達されます。センサー・プロセッサは、それ以降パケット ID を切り替える必要があります。センサー・プロセッサは、リセット後最初のパケットを受信すると、すべてのパケット ID を受け付けて、追加のパケットに備えてパケット ID を切り替えることを想定します。SmartMesh IP では、モートがリセットした場合、モートは常に同期ビットを設定した状態で起動イベントを送信します。SmartMesh WirelessHART では同期ビットは設定されませんが、センサー・プロセッサは各起動イベントを他と重複しないイベントとみなすことができるので、モートからのすべてのパケット ID を受け付けて、追加のパケットに備えてパケット ID を切り替えることを想定できます。

34.5 パケット ID の無視

センサー・プロセッサの実装を単純化してとにかく機能させるため、センサー・プロセッサは受信側のすべてのコマンドを新しいものとみなして処理することにより、実質的にパケット ID を無視できます。送信側では、センサー・プロセッサはすべてのメッセージについて同期ビットを設定し、モートがすべてのパケットを新しいコマンドとして処理することを確保する必要があります。こうすると、信頼できる呼び出し応答インターフェースの堅牢性がいくらか失われるので、試作品以外では推奨の動作ではありません。

35 アプリケーション・ノート：バックグラウンド・ノイズの監視

35.1 概要

SmartMesh IP ネットワークはネットワーク全体の帯域内ノイズを監視するため、自動的にデータを収集および発行しています。このデータを含む健全性レポートが、モートによって 15 分ごとに発行されます。モートによって使用されるすべてのチャンネルに対して、平均バックグラウンド RSSI、送信試行回数、失敗回数が収集されます。アプリケーションは、このデータを、ネットワーク全体での干渉による影響を示す仮想分散スペクトラム・アナライザとして使用できます。特定のチャンネルの性能が不十分な場合、ネットワークにブラックリストを設定してから再起動するか、またはこの情報を使用して干渉を探し出して、可能な場合は環境から取り除くことができます。

35.2 機能の詳細

この機能を使用するには、モートとマネージャの両方が以下のバージョン（以上）である必要があります。

- SmartMesh IP モート・スタック 1.4.1
- SmartMesh VManager 1.0.1 または SmartMesh IP 組込みマネージャ 1.4.1

モートは自動的に測定を行い、測定データを含む健全性レポートを送信します。マネージャは、通知を通じてこのデータを自動的に発行します。この機能を有効化するための操作は必要ありません。しかし、バックグラウンド・ノイズ測定値に基づいて、マネージャがネットワーク動作を変更することはないので、結果に応じて措置を取る責任はお客様にあります。

この機能には、バックグラウンド RSSI 測定とチャンネル安定性という 2 つのコンポーネントがあります。

35.2.1 バックグラウンド RSSI の測定

RF 信号の予測は難しいので、モートには通常使用する帯域幅よりも多くの帯域幅がプロビジョニングされています。そのため、モートのウェイクアップ中に受信リンクが割り当てられても、隣接モートのキューにはこのリンクに送信するパケットがありません。受信側から見た場合、このイベントは**アイドル・リスニング**と呼ばれます。このとき、モートは自身への伝送宛ではないリンクにロックオンし、オーバーヒア（スヌープ）をします。このとき、モートはパケットをドロップしていますが、アイドル・リスニング動作として、（パケット・エラーとして）カウントはされてません。アイドル・リスニングの最後に、毎回、モートはリスニングしていたチャンネルの RSSI を低消費電力で素早く測定します。この時点で、チャンネルにはネットワークからの 802.15.4 トラフィックがないはずです。モートがアイドル・リスニング時に測定を行うたびに、このチャンネルでのその他の測定値と合わせて RSSI の平均がとられます。特定のチャンネルでモートの近くにバックグラウンド干渉がある場合（別の SmartMesh ネットワーク、Wi-Fi ネットワーク、または 2.4 GHz 帯のデバイスからの干渉など）、モートがこのチャンネルで測定する RSSI 値が高くなることが想定されます。

35.2.2 チャンネルの安定性

モートは MAC 層のユニキャスト・メッセージを送信するたびに、受信者からアクノリッジを受け取ることを想定しています。モートがパケット送信後にアクノリッジを受け取らなかった場合、モートはこの送信を失敗としてカウントし、後でパケット送信を再試行します。チャンネルの安定性を追跡するため、モートは各チャンネルでの送信回数と失敗回数を継続的に集計します。この 2 つの回数からチャンネルの安定性を計算できます。例えば、モートがチャンネル 5 で 10 回送信を行い、2 回失敗した場合、チャンネル安定性は 80%になります。前のセクションと同様に、特定のチャンネルでモートの近くにバックグラウンド干渉がある場合、そのチャンネルのチャンネル安定性が低くなることが想定されます。

35.3 健全性レポートのパケット形式

チャンネル固有の健全性レポートには、15 チャンネル分のデータが含まれます。各パケットは、netExtHealthNotif ID で拡張健全性レポートとして分類されます。

タイプ	説明
INT8U	TLV タグ（RSSI レポートの場合は常に 0x01）
INT8U	TLV の長さ（下記のペイロードの長さ）
RSSI_REPORT_T[15]	15 チャンネル分の RSSI レポート（構成は以下を参照）

各 RSSI レポートの構成は以下のとおりです。

タイプ	説明
INT8S	当該チャンネルの平均アイドル・リスニング RSSI
INT16U	当該チャンネルでのユニキャスト試行回数
INT16U	当該チャンネルでアクノリッジされなかった回数

VManager のトレースから出力された例を次に示します。チャンネルごとに各行が出力されます。L_TRACE RX ie=32:1e mesh=4022 ttl=126 asn=19661 gr=1 src=83 rt=g tr=u:req:0; sec=s ofs=23 nc=27 ie=fe:00 IP=7e77 UDP=f7 sP=f0b0 dP=f0b0; cmd=HR len=77hr=01:4b:a5:00:10:00:01:a6:00:15:00:02:aa:00:15:00:00:a6:00:0d:00:01:ad:00:11:00:01:a7:00:19:00:01:ac:00:0f:00:04:a

このモートで測定されたバックグラウンド・ノイズのうち最も高いのはチャンネル 4 で、平均 RSSI は-82dBm（16 進値 `0xad` より）です。チャンネル 1 のチャンネル安定性を計算すると、**s** = 1-2/21 = 90.5%になります。

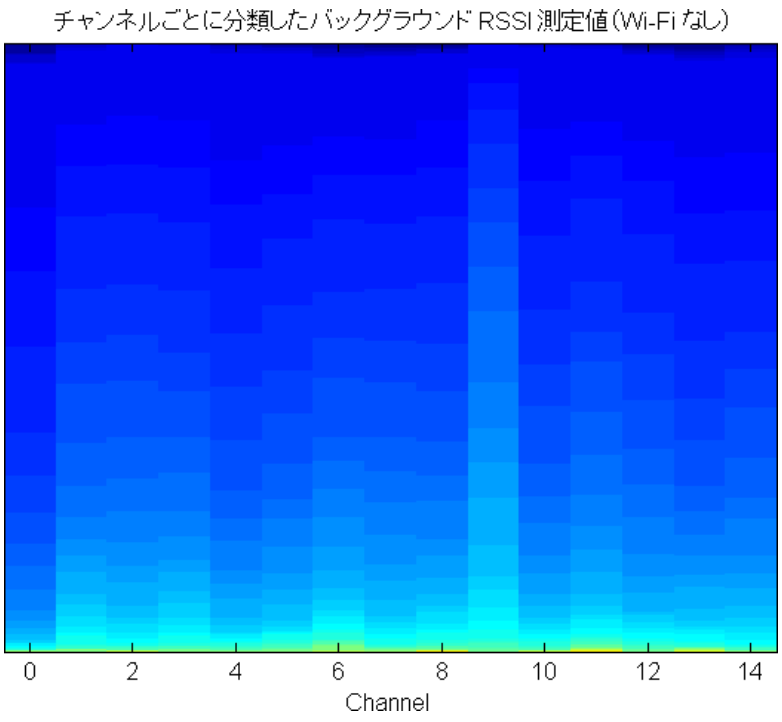
35.4 結果の解釈

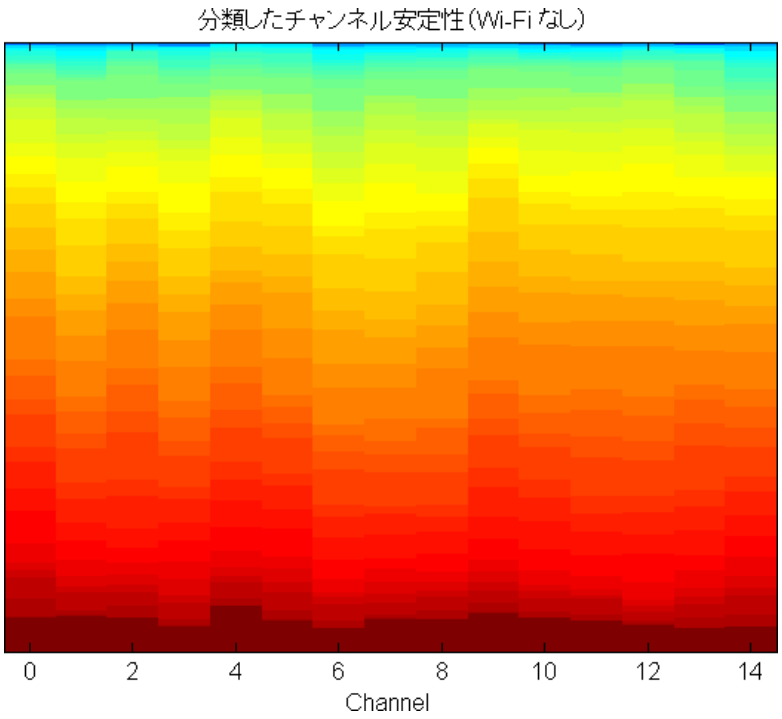
基本的に、ワイヤレス・ネットワークでのパケット送信を失敗させない限り、バックグラウンド・ノイズに注意する必要はありません。このため、チャンネル安定性の方が、バックグラウンド RSSI 測定値よりも有用な測定基準になります。ただし、バックグラウンド RSSI は干渉の位置を突き止めるヒントになる場合があります。例えば、モート A とモート B

があり、モート A が上流の B に送信しているとします。A がチャンネル 5 に対して低いチャンネル安定性を報告していても、バックグラウンド RSSI が-90dBm と低い場合、この干渉によってパケット失敗が引き起こされているとは考えられません。この場合、B から報告されたチャンネル 5 のバックグラウンド RSSI を確認します。B から報告されたバックグラウンド RSSI が高ければ(-60dBm など)、A がこのチャンネルで親への送信に失敗する原因だと考えられます。

35.4.1 干渉がない場合の結果

250 のデバイスを含むネットワークをセットアップし、64 のデバイスがバックグラウンド・ノイズ健全性レポートを送信するように設定しました。ネットワークを約 3 時間実行し、レポートを送信する 64 のモートからそれぞれ 12 個の健全性レポートを収集しました。このネットワークでは、干渉の発生源を特定するのではなく、全体的なノイズ特性を確認するために、すべての健全性レポート・データを 1 つの分布としてまとめました。チャンネルごとに、モートから報告されたすべてのバックグラウンド RSSI 値とチャンネル安定性を確認しました。その結果をまとめたものが下図であり、各列が分類された 1 チャンネル分のデータを表しています。



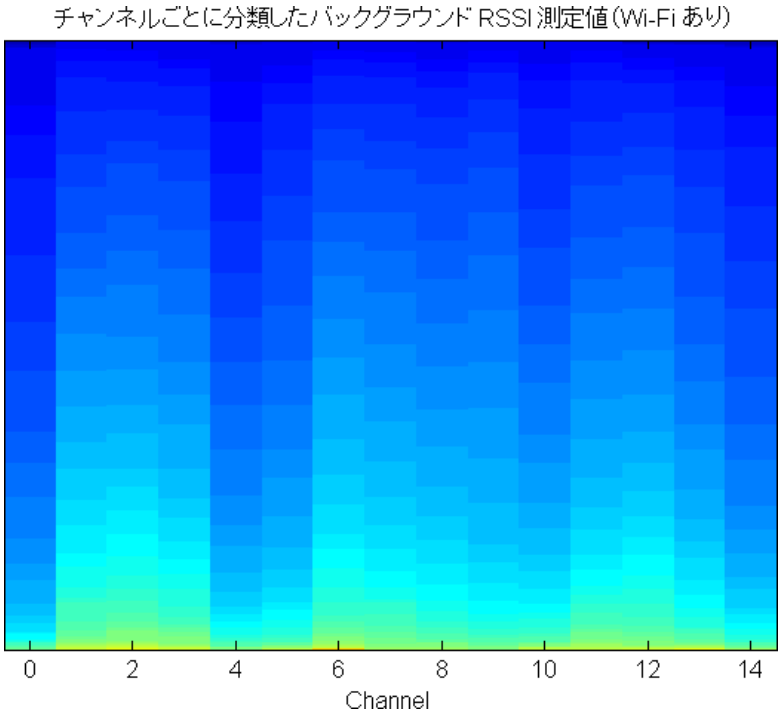


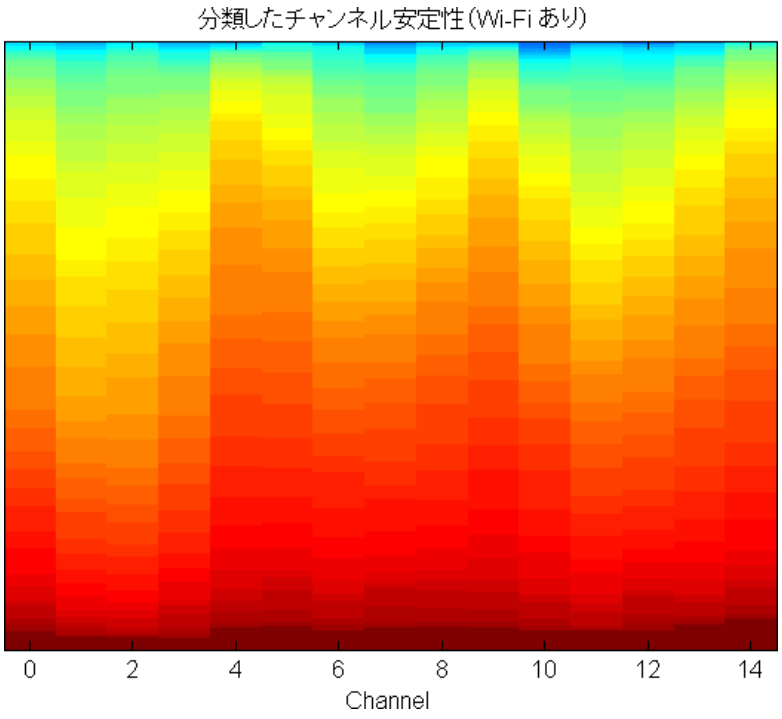
バックグラウンド RSSI では、濃い青色が、15 分間のアイドル・リスニング・イベントに対する平均測定値が-100dBm であることを表しています。これらの濃い青色は、グラフの一番上に分類されており、あまり目立ちません。濃い緑色は平均値-67dBm を表しますが、グラフ上に緑、黄色、赤はほとんどないので、ほとんどの時間、平均 RSSI が-67dBm 未満であったことがわかります。また、チャンネル 9 で少量のノイズが測定されていますが(明らかに、隣接オフィスの Wi-Fi ネットワークによる)、どのチャンネルにも大きな干渉ピークはありません。

同様の構成がチャンネル安定性のプロットにも使用されています。ここでは、濃い赤が 15 分間でのこのチャンネルの成功率 100%を示しており、濃い青色が 0%の成功率(この間隔で当該チャンネルの全試行が失敗した)を表しています。中間点となる濃い緑は 50%の安定性を表します。ここでも、すべてのチャンネルを通じて比較的均等な安定性が得られています。

35.4.2 Wi-Fi 干渉がある場合の結果

2 番目のテストでは、この建物で 2.4GHz 帯の Wi-Fi を作動させました。下記の 2 つのグラフでは、前のセクションと同じ基準で色をプロットしています。



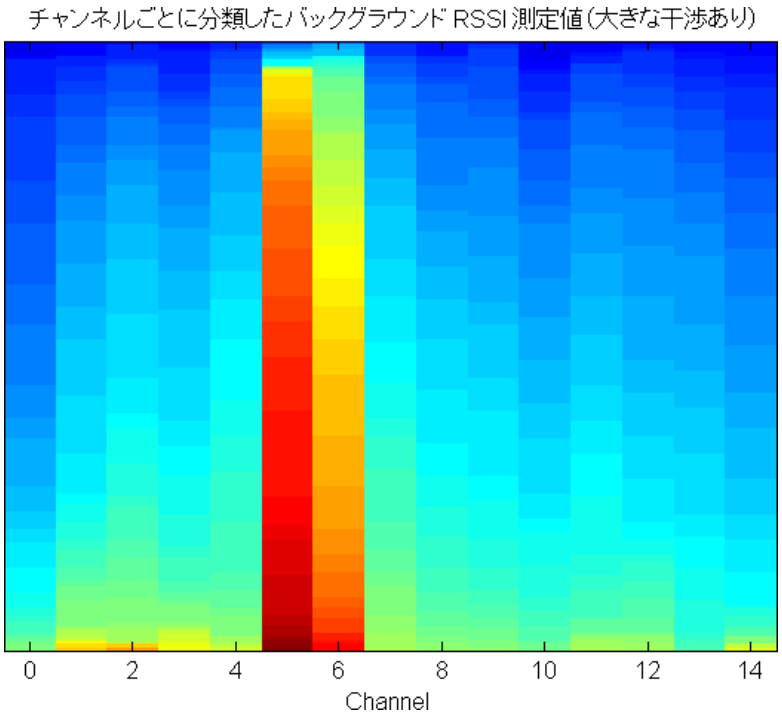


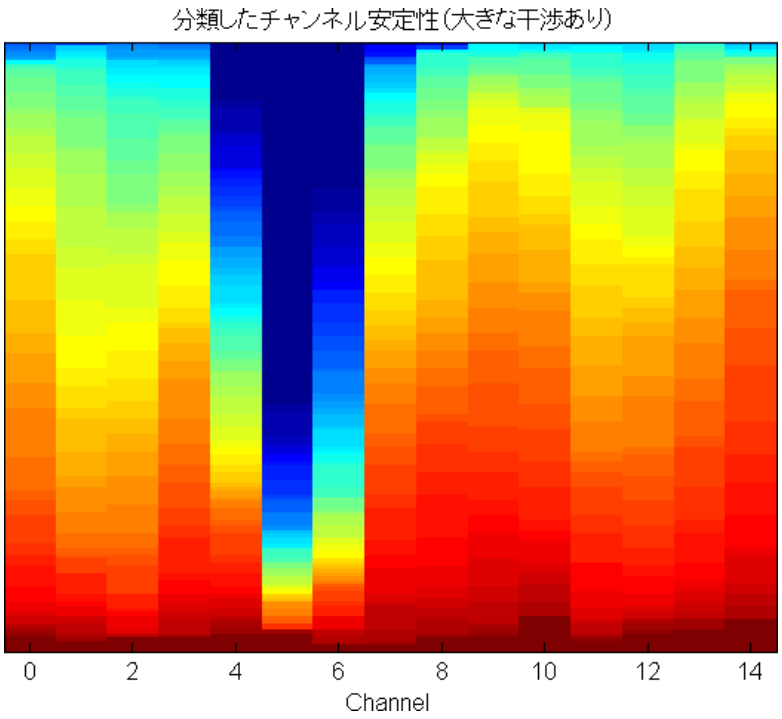
バックグラウンド RSSI のグラフでは、3 つの帯域で Wi-Fi 干渉があることが見てとれ、それぞれがだいたい 3 つのチャンネルに渡っています。ここから直感的に、Wi-Fi チャンネルの帯域幅は 802.15.4e チャンネルの約 3 倍である

と想像できます。バックグラウンド RSSI グラフに小さなピークが見られるチャンネルでは、安定性グラフでも若干の減少が見られるので、Wi-Fi が作動しているときにネットワークで送信の失敗が引き起こされていることが連想されます。良い点は、Wi-Fi の追加によって低下したネットワーク全体の安定性はわずか 4%であり、総合的な影響は最小限に抑えられていることです。SmartMesh ネットワークは自動的に干渉を避けてホッピングするように設定されており、これがまさにその例になります。経験上、モートで測定されたバックグラウンド RSSI が一部のチャンネルに限定されており、安定性に大きな影響がない場合、軽減措置を講じる必要はありません。

35.4.3 一定の狭帯域干渉がある場合の結果

3 番目のテストでは、大きな GSM 符号化干渉を連続的に 7MHz 帯域幅に送出しました。このシステムの送信出力は Wi-Fi アクセス・ポイントと同じでしたが、Wi-Fi とは違い、この干渉は周期化されていませんでした。このため、下記のグラフに示すように、より深刻な影響が 2 つのチャンネルに及びました。





チャンネル 5 には重大な影響があり、隣接チャンネルでも大幅に安定性が低下しています。この干渉は非常に大きいので、チャンネル 5 を使用するネットワーク内の伝送はほぼすべて失敗すると考えられます。

35.4.4 結果に基づくブラックリスト登録

干渉が少数のチャンネルに限定されていても、ネットワーク全体に広く行き渡っている場合、**PUT /network/config** コマンド(VManager)または **setNetworkConfig** コマンド(組込みマネージャ)を使用して、ブラックリストを設定することを検討してください。ブラックリストの登録については、いくつかの注意点があります。まず、ブラックリストの設定にはネットワークのリセットが必要です。次に、チャンネルをブラックリストに登録した後、モートは登録されたチャンネルのバックグラウンド RSSI を監視することも、チャンネル安定性を測定することもできないので、干渉パターンの変化を見極めることが難しくなります。3 番目に、ブラックリストはネットワーク全体に適用されるため、ネットワークの部分ごとに異なる干渉が発生している場合は役に立ちません。ブラックリストをネットワークに設定したら、その前後での安定性を比較します。安定性が改善されている場合、ネットワークの性能が向上し、待ち時間と平均モート消費電力が両方とも減少します。上記のサンプル結果については、Wi-Fi のみのケースではブラックリストの登録を推奨しませんが、狭帯域干渉のケースでは、チャンネル 4、5、6 をブラックリストに登録することを推奨します。安定性が33%を下回るチャンネルがある場合、ブラックリスト登録によって特に待ち時間と信頼性が改善する可能性があります。

干渉が全帯域およびネットワーク全体に影響している場合、唯一の解決策は、ネットワーク内にあるデバイスの送信出力を上げることです。具体的には、無線回路にパワー・アンプを追加し、各国の規制で許可されているレベルまで出力を上げます。受信側のアンプでリンク・バジェットを増加した場合、必要な信号と同じだけノイズも増幅され、SNR が変わらないので、この方法は有効ではありません。これが不可能な場合、ネットワーク内のプロビジョニングを大きくして生成パケットあたりの試行回数を増やすか、経路安定性の不十分な 2 つのデバイス間に中継器を設置します。

干渉の場所を限定できる場合、影響のあるモートの位置を使用して干渉発生源を特定することができます。場合によっては、干渉物を除去したり、その出力を弱めることで、ネットワーク性能を改善できる可能性もあります。

以前に、非常に大きい帯域外干渉（WiMax など）によって、帯域の端で安定性の問題が発生しているケースがありました。帯域内の RSSI の直接測定ではまったくエネルギーが測定されない場合でも、チャンネル安定性の測定値を使用して干渉の発生源を見つけることができます。

商標

Eterna、Mote-on-Chip、SmartMesh IP は、Dust Networks, Inc の商標です。Dust Networks ロゴ、Dust、Dust Networks、SmartMesh は、Dust Networks, Inc の登録商標です。LT、LTC、LTM、▶ は、アナログ・デバイセズの登録商標です。第三者のブランド名および製品名は各社の商標であり、情報提供のみを目的として使用されています。

著作権

免責事項

本書は現状のまま提供され、明示、暗示を問わず一切の保証を行なわないものとします。かかる保証には、特定目的に対する商品性または適合性の黙示的保証が含まれますが、これに限定されません。

本書には技術的に不正確な記述またはその他の誤りが含まれる場合があります。訂正と改善は、新しいバージョンの文書に取り入れられる可能性があります。

アナログ･デバイセズは、製品やサービスの適用または使用により発生する責任を負いません。また、間接的あるいは偶発的損害を含むがそれに限定されない、いかなる責任も負わないものとします。

アナログ・デバイセズの製品は、誤動作が深刻な人身傷害につながると合理的に予想できる生命維持装置、デバイス、またはその他のシステムでの使用、またはその機能不全により生命維持装置またはシステムの故障あるいはその安全性や有効性に影響すると合理的に予想できる生命維持装置またはシステムの重要な部品としての使用を目的として設計されていません。このような用途での使用を目的としてこれらの製品を使用または販売しているアナログ・デバイセズの顧客は、顧客自身の責任でそれを行い、このような意図しないまたは不正な使用に関連する人身傷害または死亡に直接または間接的に起因するすべての主張、費用、損害、支出、および妥当な額の弁護士費用、また、かかるクレームでアナログ・デバイセズに該当製品の設計または製造に関わる過失があったと主張される場合でも、これを完全に補償し、アナログ・デバイセズとその役員、従業員、子会社、関連会社、および販売代理店に何ら損害を与えないことに同意するものとします。

アナログ・デバイセズは、いつでも製品またはサービスに対する修正、変更、拡張、改良、その他の変更を行う権利を保有し、製品またはサービスを予告なく中止する権利を有します。顧客は、発注の前に最新の関連情報を入手し、その情報が最新で完全であることを確認する必要があります。すべての製品は、注文承諾時または販売時に提供される、販売に関する Dust Network の契約条件に従い販売されます。

アナログ・デバイセズは、アナログ・デバイセズの製品またはサービスが使用される組み合わせ、マシン、またはプロセスに関連するアナログ・デバイセズの特許、著作権、マスクワーク、その他のアナログ・デバイセズの知的所有権に従って、明示か黙示かにかかわらず、ライセンスが付与されることを保証または主張するものではありません。第三者の製品またはサービスに関してアナログ・デバイセズが公開した情報は、その製品またはサービスを使用するためのアナログ・デバイセズからのライセンス提供、あるいはその保証または推奨を意味するものではありません。このような情報を使用する場合、第三者の特許または他の知的所有権に従って第三者からのライセンスが必要になるか、またはアナログ・デバイセズの特許または他の知的所有権に従ってアナログ・デバイセズからのライセンスが必要になります。

Dust Networks, Inc は、アナログ・デバイセズの完全所有子会社です。