イーサネット 生フレーム通信 (2)（公開資料） (Ethernet raw frame communication (2))	作成 2022/4/5 ニューラルソフト有限会社 市来 博記

初版	－	2022/4/5

概要

本書は、「イーサネット 生フレーム通信 (1)」に記載できなかった 2 台の Raspberry Pi Model 4B の内蔵イーサネットを相互に直接接続したネットワークにおける通信シーケンスの応答時間の測定結果を記したものです。（通信シーケンスは図 2 と図 3 を参照のこと。）

調査内容

イーサネット 生フレーム通信に関する調査項目は、「イーサネット 生フレーム通信 (1)」を参照のこと。

情報源

情報源は、「イーサネット 生フレーム通信 (1)」を参照のこと。

関連情報

Raspberry Pi リモート開発環境の構築
イーサネット 生フレーム通信 (1)

前提条件

調査における前提条件を示します。

- 調査対象の通信環境は、2 台の Raspberry Pi Model 4B の内蔵イーサネットを相互に直接接続したネットワークとします。
- Raspberry Pi に搭載する OS は、以下の通りとします。
  Model 4B(Mem 8GB)：Raspberry Pi OS 11 Bullseye 64BIT
  Model 4B(Mem 4GB)：Raspberry Pi OS 11 Bullseye 64BIT
- 調査用プログラムの作成と動作確認は Visual Stdudio 2022 Community（C++による Linux 開発）を使用します。（Visual Stdudio 2022 を使用した Raspberry Pi Model 4B のリモート開発環境は構築済みとします。構築方法は、「Raspberry Pi リモート開発環境の構築」を参照して下さい。）
- 開発用ホスト マシンは、Windows10（64 ビット）がインストールされた PC（i7-3930K，32GB メモリ搭載の PC）を使用します。
  開発/調査環境は図 1 の通りです。

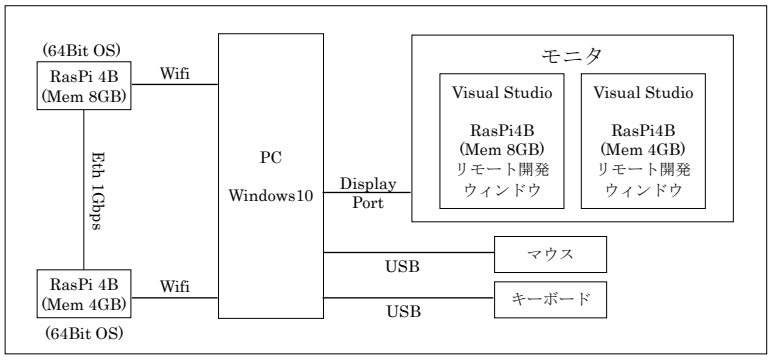

図 1 開発/調査環境

- イーサネットの生フレームの送受信は、ソケット API（RAW モード）を使用して実現します。
- 本書の記載内容は、C/C++言語での UDP ソケット通信プログラム作成の経験がある方が対象です。

調査結果

〔通信シーケンスにおける応答時間〕以外の項目に関する調査結果は「イーサネット 生フレーム通信 (1)」を参照のこと。

〔通信シーケンスにおける応答時間〕

応答時間測定の対象となる通信シーケンスの概要を図 2 に示します。

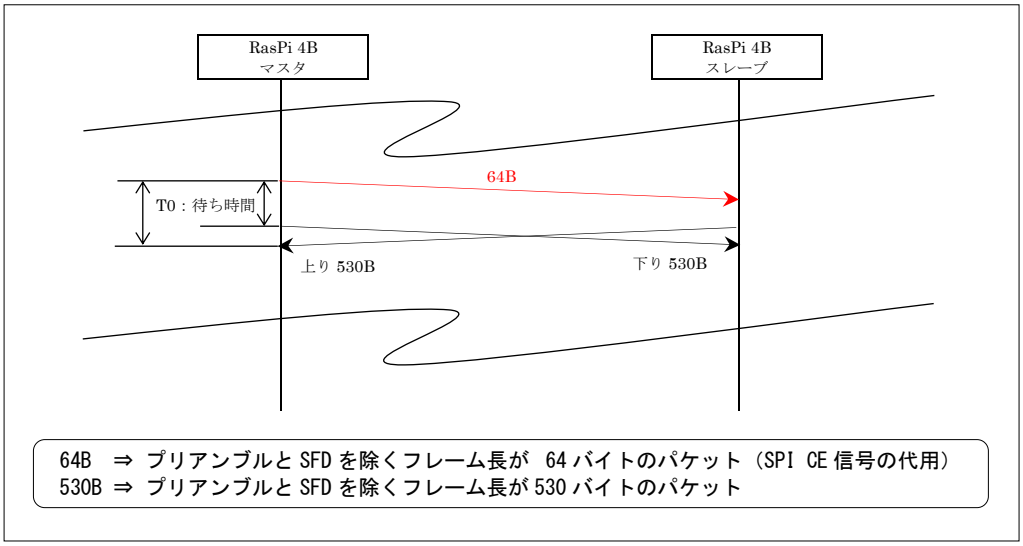

図 2 応答時間測定の対象となる通信シーケンスの概要

上記のシーケンスは、64B をトリガとして、マスタとスレーブで 530B を全二重で送受信する単純なシーケンスですが、色々な要因があり、きれいな全二重通信になりません。（T0 が一定にならない為）
実際のシーケンスを図 3 に示します。（推測を含みます）

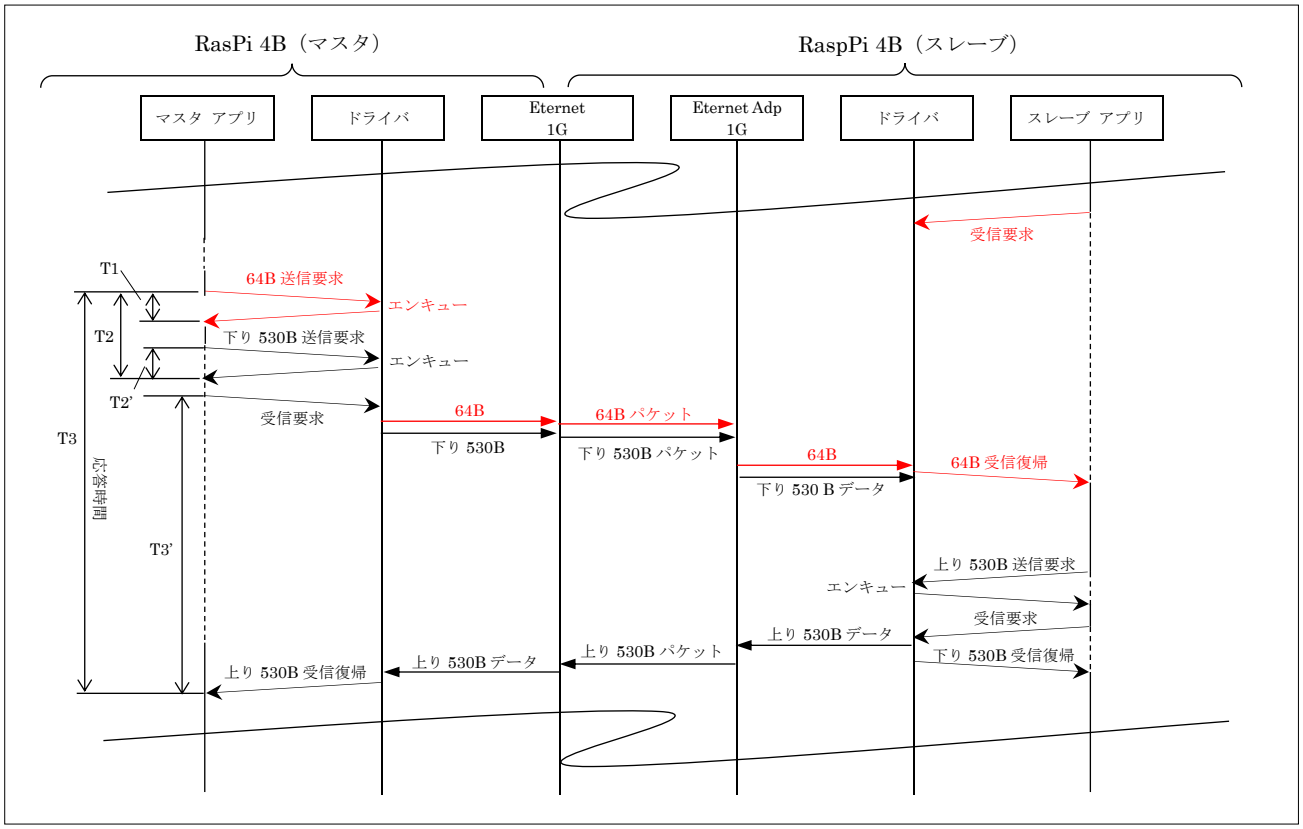

図 3 応答時間測定の対象となる通信シーケンス

上記のシーケンスを約 4 ミリ秒周期で 1000 回繰り返して応答時間を計測（clock_gettime 関数を使用）した結果を以下に示します。

〔計測の共通設定〕

CPU の周波数設定	cpufreq-set -g コマンドで performance を設定 （cpufreq-info -f コマンドで 1500MHz であることを確認）
eth0 の省電力の設定	sudo ethtool --set-eee eth0 eee off コマンドでオフ （ethtool --show-eee eth0 コマンドで Power Management off を確認）
eth0 の割込みモデレーション	デフォルト設定 RX : 1 パケット受信完了毎に受信完了割り込みが発生 adaptive-rx : off, rx-usecs : 57, rx-frames : 1 受信完了パケット数が rx-frames に達するか、最初のパケットを受信完了時点から rx-usecs 経過した場合に受信完了割り込みが発生 TX : 10 パケット送信完了毎に受信完了割り込みが発生 adaptive-tx : n/a（設定不可） tx-usecs : n/a tx-frames : 10（1 に設定しても計測結果に影響がなかった）
wlan0 の省電力の設定	sudo iw dev wlan0 set power_save off コマンドでオフ （iwconfig wlan0 コマンドで Power Management off を確認）
計測プログラムのスレッド数	シングル
プロセスの優先度	-10
スレッドのスケジュール設定	ポリシー、優先度ともに変更なし
T0	0

〔計測 1：sendto と recvfrom 関数を使用した場合〕

条件

送信タイムアウト値：100 マイクロ秒
送信タイムアウト時の動作：1 回でもタイムアウトが発生した場合は、計測を異常終了する。
受信タイムアウト： マスタ側：1 ミリ秒／スレーブ側：5 ミリ秒
受信タイムアウト時の動作：
スレーブ側の最初の 64B 受信：無限に受信を繰り返す。
その他：1 回でもタイムアウトが発生した場合は、計測を異常終了する。

結果

```
        Ave,   Max,   Min
t1 =     7,     19,     7[usec](Over30 = 0) (Ave と Min の差 0[usec])
t2 =    13,     26,    12[usec](Over30 = 0) (Ave と Min の差 1[usec])
t3 =    71,    148,    67[usec]  (Ave と Min の差 4[usec])
```

この結果は、ping コマンドによる 64 バイト データ折り返しにかかる時間と近い値になっています。

```
PING XXX.XXX.XXX.XXX (XXX.XXX.XXX.XXX) 56(84) bytes of data.
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=1 ttl=64 time=0.106 ms
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=3 ttl=64 time=0.069 ms
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=4 ttl=64 time=0.063 ms
                              :
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=29 ttl=64 time=0.061 ms
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=30 ttl=64 time=0.049 ms
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=31 ttl=64 time=0.056 ms
64 bytes from XXX.XXX.XXX.XXX: icmp_seq=32 ttl=64 time=0.053 ms
                              :
```

図 2 の理想的な全二重通信が行われた場合、ギガ ビット イーサネットの電気信号（(8+64+12+8+530)×8Bit）の伝送にかかる時間は約 5μ秒なので、単発パケットの送受信のオーバー ヘッドは送信側と受信側合わせて約 33 マイクロ秒程度と思われます。

〔計測 2：64B を GPIO の 1 ビットのデジタル信号に置き換えた場合(1)〕

条件

図 3 のシーケンスの赤い矢印の流れを GPIO の 1 ビットのデジタル信号に置き換える。
スレーブ側は、立ち上がりエッジ イベントをタイム アウトを指定して待ち受ける。(必ず計測タスクは I/O 完了待ち状態になる。)

結果

```
        Ave,   Max,   Min
t1 =     2,     5,     2[usec](Over30 = 0) (Ave と Min の差 0[usec])
t2 =    10,    25,     9[usec](Over30 = 0) (Ave と Min の差 1[usec])
t3 =    58,   119,    55[usec] (Ave と Min の差 3[usec])
```

計測 1 の結果から約 12 マイクロ秒短縮しました。

〔計測 3：64B を GPIO の 1 ビットのデジタル信号に置き換えた場合(2)〕

条件

図 3 のシーケンスの赤い矢印の流れを GPIO の 1 ビットのデジタル信号に置き換える。
スレーブ側は、デジタル信号の変化を繰り返し読み出しで検出する。(なるべく計測タスクが I/O 完了待ち状態にならないようにする。)

結果

```
        Ave,   Max,   Min
t1 =     2,     5,     2[usec](Over30 = 0) (Ave と Min の差 0[usec])
t2 =     9,    25,     9[usec](Over30 = 0) (Ave と Min の差 0[usec])
t3 =    38,   103,    28[usec] (Ave と Min の差 10[usec])
```

計測 1 の結果から約 33 マイクロ秒、計測 2 の結果から約 20 マイクロ秒短縮しました。

〔計測の揺らぎ〕

1～3 の計測を複数回行うと結果に揺らぎが発生します。最小値と平均値は数マイクロ秒の範囲ですが、最大値は 100 マイクロ秒程度になる場合があります。揺らぎを小さくするチューニング（不要ハードウェアーやソフトウェアー モジュールの無効化）については別件として調査します。

所感

イーサネット 生フレーム通信 (1)の調査と計測の条件を変更した為、一概には言えませんが、Raspberry Pi 4 のイーサネットの性能は Raspberry Pi 3/3+から大きく改善されたと言えると思います。ですが、単発パケットの送受信で発生するオーバー ヘッドが大きいような気がします。(正直な所、計測結果の半分位の時間を予想していました。チューニングでオーバー ヘッドを小さくできるかも知れません。)
ロボット制御などで Raspberry Pi 4 + Raspberry Pi OS 11 Bullseye 64BIT を採用する場合、ハードウェアーの割り込みが重複しないようなシーケンスやパイプライン処理などの工夫が必要になると思われます。