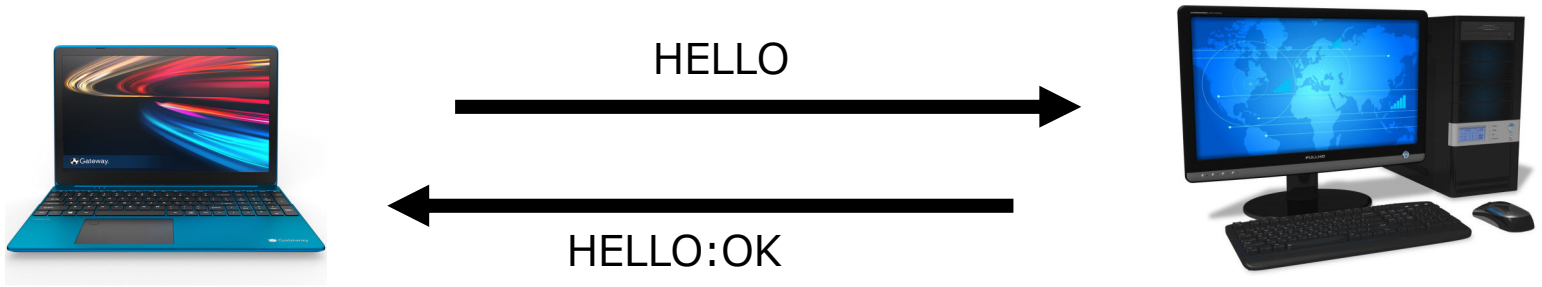
TCPBenchMark

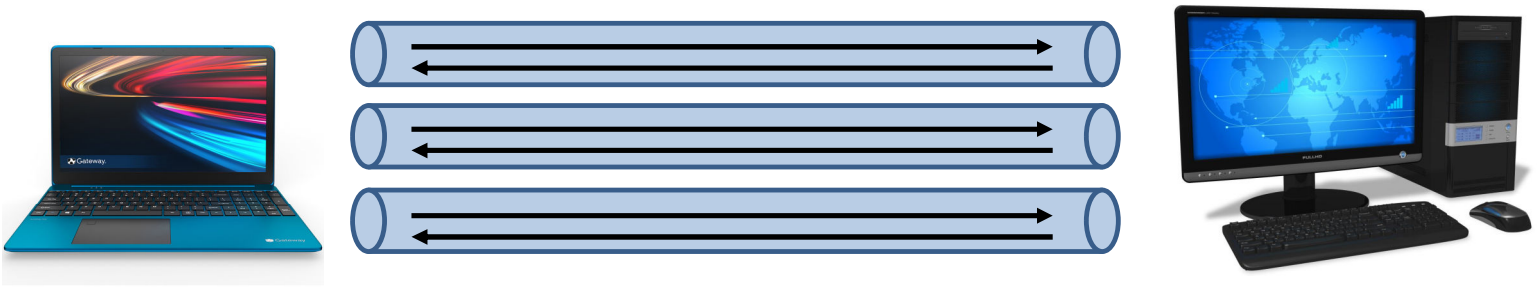
Echo Back サーバに対するベンチマークプログラム

- ベンチマーク対象のサーバ仕様
  - TCP/IP接続を受け付ける
  - 特定のポート番号への接続を受け付ける
  - 送信メッセージをそのまま応答する（あるいは特定の文字列をメッセージの後ろに付与して応答する）
    - 例: “HELLO”を送信したら”HELLO”を応答する
    - 例: “HELLO”を送信したら”HELLO:OK”を応答する



ベンチマークプログラムの概要

- マルチスレッドでサーバに対して同時アクセスを行います．
- 指定可能なパラメータは以下の通り
  - スレッド数
  - 各スレッドで送信するメッセージのサイズ
    - 指定サイズのランダム文字列を生成します
  - 各スレッドのEchoBack確認回数
- EchoBackメッセージの内容チェック
  - 送信メッセージと一致するか，既定のポストフィックスを加えたものとします（ポストフィックスはコード中で指定）．
- スレッド単位で連続したEchoBack確認回数分成功した場合のみ成功と判定します．
  - 途中で送受信失敗があった場合，そのスレッドは接続からやり直します．



ベンチマークプログラムの実行例

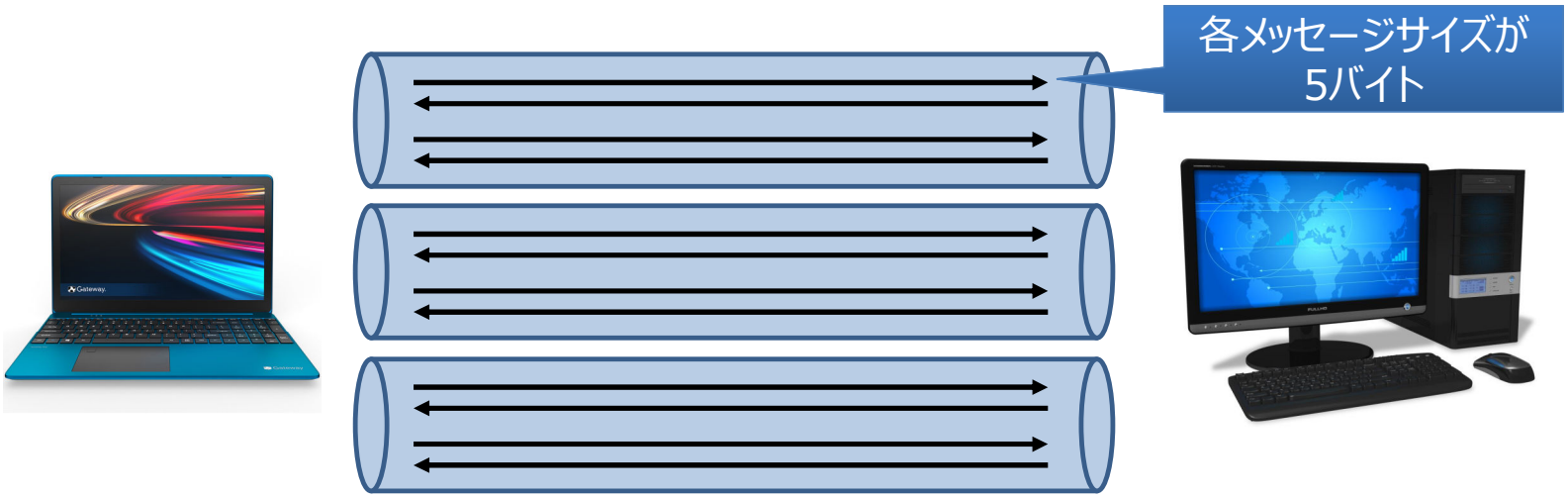
- 以下の実行例では，以下のパラメータが指定されています．
  ① 接続先ホスト: 192.168.0.185
  ② 接続先ポート番号: 10000
  ③ スレッド数: 3
  ④ メッセージサイズ: 5 (Byte)
  ⑤ メッセージ送信回数: 2回（スレッドあたり）
- 即ち，「192.168.0.185の10000番ポートに3スレッド同時アクセスを行い，各スレッドでは『5バイトのメッセージを送信し，その応答を受け取る』ことを2回実施する」ことになります．

```
$ ./tcpbenchmark 192.168.0.185 10000 3 5 2
```

ベンチマークプログラムの実行例

- 「192.168.0.185の10000番ポートに3スレッド同時アクセスを行い，各スレッドでは『5バイトのメッセージを送信し，その応答を受け取る』ことを2回実施する」ことになります

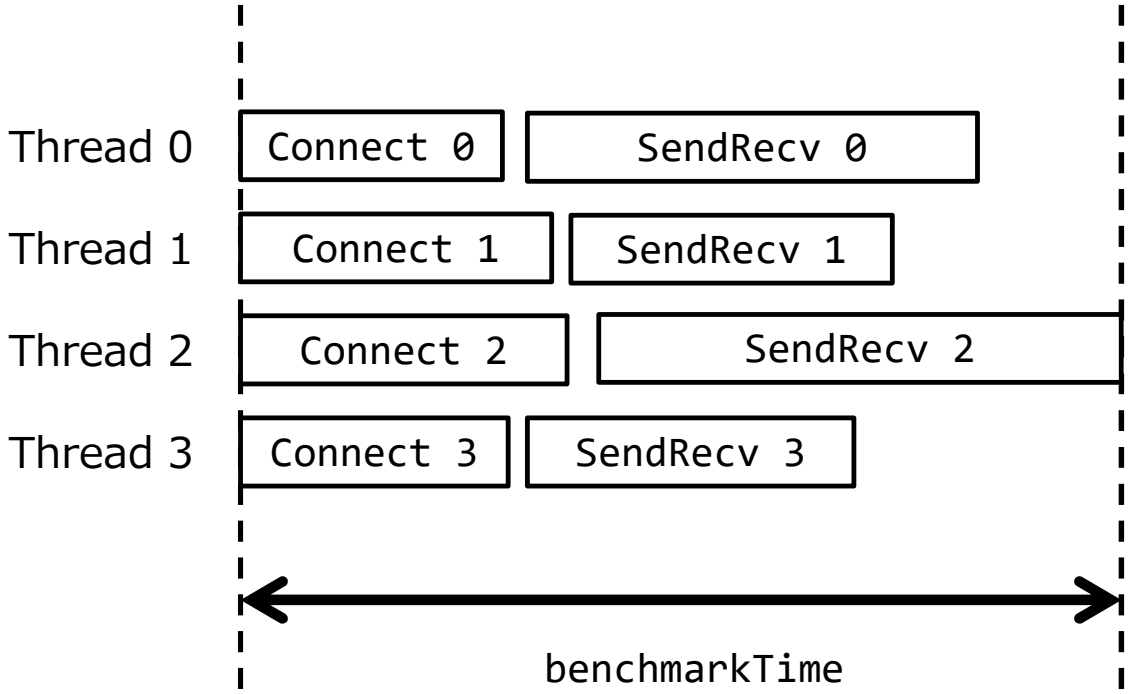
```
$ ./tcpbenchmark 192.168.0.185 10000 3 5 2
```



ベンチマークプログラムの実行結果

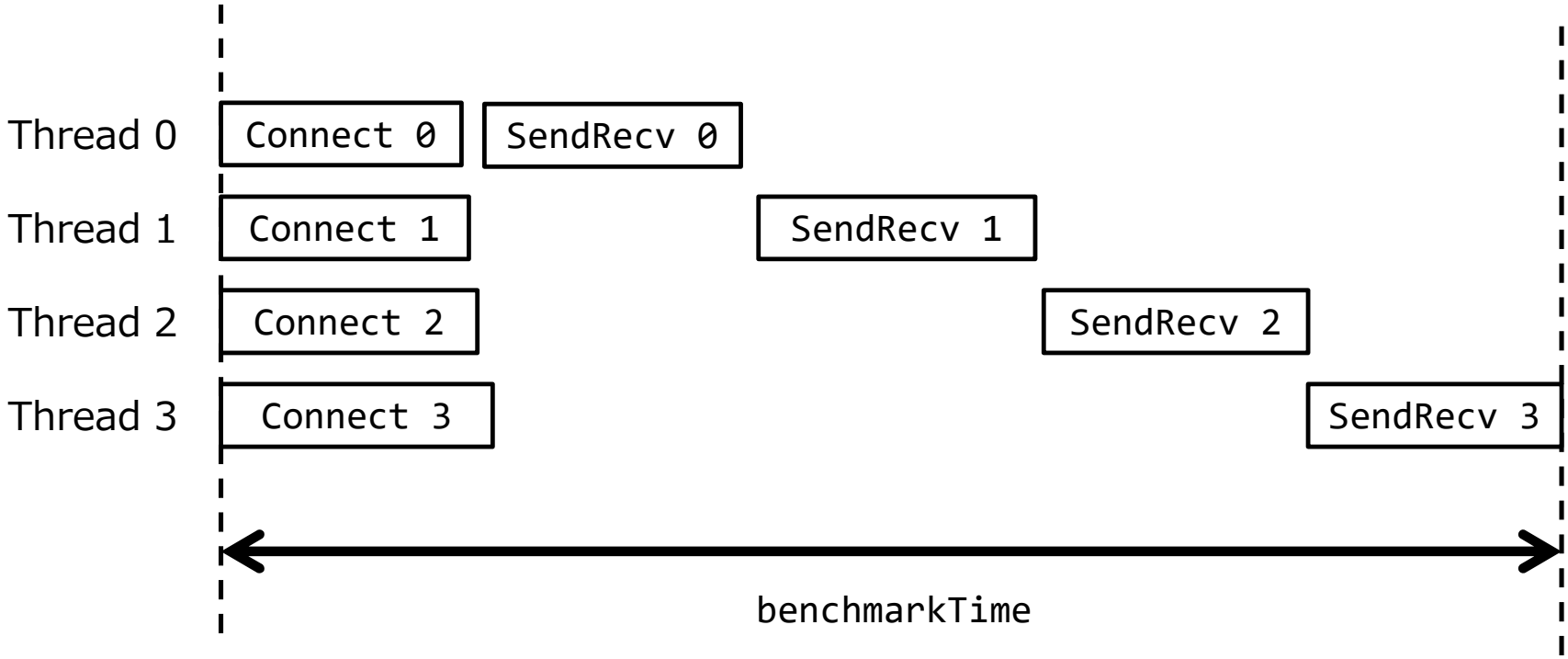
カテゴリ	タイプ	概要
	thread	スレッド数（パラメータで指定した値と一致）
成功・失敗	success	パラメータ指定通りに処理が完了したスレッド数
	failedConnect	接続に失敗した数
	failedSendRecvLoop	送受信ループ（連続したN回の送受信）が正常終了しなかった数
	failedSendRecv	送受信に失敗した数
接続時間	connectTime(total)	接続に要した時間（失敗時を含みません）
	connectTime(average)	接続に要した時間の平均値
	connectTime(sample variance)	接続に要した時間の分散
	connectTime(max)	接続に要した時間の最大値
送受信時間	sendRecvTime(total)	送受信に要した時間（失敗時を含みません）
	sendRecvTime(average)	送受信に要した時間の平均値
	sendRecvTime(sample variance)	送受信に要した時間の分散
	sendRecvTime(total)	送受信に要した時間の最大値
	benchmarkTime(include failed)	ベンチマークが終了するまでの時間（失敗時を含みます）

ベンチマークプログラムの計測概要（例）




```
connectTime(total) = Connect 0 + Connect 1 + Connect 2 + Connect 3
sendRecvTime(total) = SendRecv 0 + SendRecv 1 + SendRecv 2 + SendRecv 3
```

ベンチマークプログラムの計測概要（例）




```
connectTime(total) = Connect 0 + Connect 1 + Connect 2 + Connect 3
sendRecvTime(total) = SendRecv 0 + SendRecv 1 + SendRecv 2 + SendRecv 3
```

ベンチマークプログラムの実行結果例（接続開始まで）

```
$ ./tcpbenchmark localhost 10000 3 5 2
hostname = localhost, port = 10000          # パラメータ確認
3 thread with 2 echoback.                   # パラメータ確認
message size is 5 bytes.                    # パラメータ確認
Thread stack size = 2097152 bytes           # 各スレッドのスタックサイズの確認
thread 0 created.                           # スレッド作成
thread 1 created.                           #    :
thread 2 created.                           #    :
start count down: 2                         # スレッドを同時開始するためのカウントダウン
start count down: 1                         #    :
Thread Start!!                              # 全スレッド開始
```

ベンチマークプログラムの実行結果例（接続）

```
addr=127.0.0.1                    # 各スレッドの接続
port=10000
addr=127.0.0.1
port=10000
addr=127.0.0.1
port=10000
```

ベンチマークプログラムの実行結果例（各スレッドの結果）

```
Thread(0): true                                          # スレッド0の結果 → 成功
  connectTime: UsedTime: 0.0013909540(sec)               # 接続処理に要した時間（成功時）
  sendRecvTime: UsedTime: 0.0003629950(sec)              # 送受信処理に要した時間（成功時）
  failedConnectNum = 0                                   # 接続失敗なし
  failedSendRecvLoopNum = 0                              # 送受信ループの失敗なし
  failedSendRecvNum = 0                                  # 送受信の失敗なし
Thread(1): true
  connectTime: UsedTime: 0.0012880700(sec)
  sendRecvTime: UsedTime: 0.0004741430(sec)
  failedConnectNum = 0
  failedSendRecvLoopNum = 0
  failedSendRecvNum = 0
Thread(2): true
  connectTime: UsedTime: 0.0016464540(sec)
  sendRecvTime: UsedTime: 0.0004576960(sec)
  failedConnectNum = 0
  failedSendRecvLoopNum = 0
  failedSendRecvNum = 0
```

ベンチマークプログラムの実行結果例（結果）

CSVの書式になっている

```
-------------------------------------------------------
thread,success,failedConnect,failedSendRecvLoop,failedSendRecv,connectTime(total),connectTime(ave
rage),connectTime(sample
variance),connectTime(max),sendRecvTime(total),sendRecvTime(average),sendRecvTime(sample
variance),sendRecvTime(max),benchmarkTime(include failed)
3,3,0,0,0,0.004325,0.001442,0.000000,0.001646,0.001295,0.000432,0.000000,0.000474,0.002886
-------------------------------------------------------
```

- 3スレッド実行して3スレッド成功
- 接続・送受信ループ・送受信ともに失敗なしなので，想定するすべての通信において失敗なし
- 接続時間: 0.004325 (sec)
- 同平均: 0.001442 (sec)，同分散は0.000000 (sec)なので，各接続処理の処理時間に揺らぎはほぼ存在せず，約0.0014 (sec)で接続処理に成功した（最大所要時間: 0.001646 (sec)）
- 送受信時間: 0.001295 (sec)
- 同平均: 0.000432 (sec), 同分散: 0.000000 (sec)なので，各送受信処理（今回は2回EchoBack）の処理時間に揺らぎはほぼ存在せず，約0.000432 (sec)で送受信処理に成功した（最大所要時間: 0.000474 (sec)）
- ベンチマーク処理全体の処理時間は 0.002886 (sec) を要した

※ benchMarkTime (0.002886) << connectTime (0.004325) + sendRecvTime (0.001295) となっているのは，benchMarkTimeはベンチマークプログラム全体（メインスレッド）で計測開始～計測終了を計測しているのに対して，connectTime/sendRecvTimeは各サブスレッド単位で計測してその和を取っているため

EchoBack応答のカスタマイズ

- EchoBack応答に特定のPostfixを付与するサーバの場合，プログラム中の以下の箇所を修正して対応することができる.

TCPBenchMark.c:

```
// 応答メッセージのポストフィックス期待値
// const char* RESPONSE_POSTFIX=":OK";
const char* RESPONSE_POSTFIX="";
```