システムプログラミング実験
ネットワーク 実験指導書

中嶋 大志 *

2024 年 12 月 18 日

1 はじめに

本実験の目的は，現在インターネットの標準となっている Transmission Control Protocol/Internet Protocol (TCP/IP) [1] の動作を実際にプログラムを作成することで学ぶことである．このために 2 つの課題と，それらに取り組むために必要な知識・技能に関する予備知識が用意されている．本実験では課題 1 まで終えることで単位を与えるものと定めている．余裕のあるものは課題 2 まで取り組むこと．課題 1 は本実験の本題で，User Datagram Protocol (UDP) [2]，TCP[3] を用いたネットワーク通信を試みる．課題 2 は実用的な雑談アプリケーションを作ることで，発展的な技術を体得することを目指す．

配布資料一覧

- `instruction.pdf`: 本資料
- `sup.pdf`: TCP/IP について説明した補助資料
- `group.pdf`: 班分けが記載された補助資料
- `udp_time_server.cpp`: UDP による時刻応答サーバのソースコード
- `udp_time_client.cpp`: UDP による時刻応答サーバに対応するクライアントのソースコード
- `tcp_time_server.cpp`: TCP による時刻応答サーバのソースコード
- `tcp_time_client.cpp`: TCP による時刻応答サーバに対応するクライアントのソースコード
- `ex_gettimeofday.cpp`: 時間計測の参考プログラム
- `command.client.cpp`: 文字列符号化方式でサーバを操作するクライアントの例．課題 2 で参考にする．
- `command.server.cpp`: 文字列符号化方式のサーバの例
- `Makefile`: 上記ファイル群のコンパイルを自動実行するためのルールを記述したファイル．

2 実験の進め方

本実験のプログラムは C++ [5] により作成する．本実験では生のデータ長を意識したプログラミングが必要である．そのため，ポインタや文字列が計算機上でどのように表現されるか復習することが望ましい．例えば，「あ」という文字を表現するために char 型変数が 2 つ必要なことは習得しておくことが望ましい．

課題提出方法，評価基準，その他注意事項は第 6 節を確認せよ．「予備知識」は各自で行い，ここでの結果は特に報告書へ記載する必要はない．課題 1 以降は 3 人，もしくは 2 人一組で行う．進捗が遅いものがいた場合は班員同士で互いに指導を行い，進捗を揃える．また，ソースコードの共有は認めるが，実験結果は各自で記録すること．例えば，サーバ側，クライアント側どちらも自ら動作させて実験を記録しなければならない．

* ver 1.0 2019 年 10 月 1 日 初版, ver 2.5 2020 年 12 月 14 日 改訂, ver 2.6 2021 年 10 月 18 日 改訂, ver 2.7 2022 年 10 月 10 日 改訂, ver 3.0 2023 年 10 月 10 日 改訂, ver 3.1 2024 年 12 月 18 日 改訂.

3 予備知識

3.1 基礎用語

以下に示す Internet protocol suite [6] の各層と Open Systems Interconnection (OSI) [9] モデルの 物理層の役割について説明できることが望ましい．

1. アプリケーション層 (Application Layer)
2. トランスポート層 (Transport Layer)
3. インターネット層 (Internet Layer)
4. リンク層 (Link Layer)
5. 物理層 (Physical Layer)

次の事柄についても，説明できることが望ましい．

1. クライアント・サーバモデル
2. Dynamic Host Configuration Protocol (DHCP)
3. ルータ（IP パケットを扱うもの）
4. スイッチングハブ（IP パケットを扱うもの）
5. デフォルトゲートウェイ（上述したルータと関係があるもの）
6. Time to Live [10]

詳細は別添の資料 `sup.pdf` 等も参考にすること．

3.2 Operating System の操作

ここでは Bash [7] を使う．ファイルの損失を防ぐために，ファイル作成などの課題はホームフォルダで行わず，適当な作業階層を作ってその中で行うことを推奨する．

3.2.1 Bash の復習

1. `dir` から始まる任意の文字列のディレクトリ（例えば，`dir1`, `dira` など）を 5 つ作成せよ
2. 手順 1 の結果を `ls` コマンドにより確認せよ．
3. `rm -r dir*` コマンドを実行し，その結果を `ls` コマンドにより確認せよ．

3.2.2 C++ の復習

1. 標準入力から 1 個の整数 n を読み取り，n 個のファイル `a1.txt`, `a2.txt`, ..., an`.txt` を出力する C++ プログラムを作成せよ．ただし，各ファイルの中身にはファイル名と同じ文字列を記載すること．
2. 手順 1 でファイルが作成されていることを `ls` コマンドにより確認せよ．
3. `rm a*.txt` コマンドによりそれらを一括削除し，その結果を `ls` コマンドにより確認せよ．

3.2.3 Bash の繰り返し

以下に示す繰り返し文を改変し，前述のプログラムにおいて $n = 10$ の場合の処理を行う Bash スクリプトを作成する．以下の文は一行ずつ打っていくと，最後に `done` を入力された時点でコマンドが実行される．

```
for i in $(seq 10)
do
echo $i.txt >> $i.txt # ファイル$i.txt に文字列"$i.txt"を書き込む命令
done
```

3.2.4 端末の IP の確認

端末をネットワークに接続後，DHCP により IP アドレスが割り振られていることをコマンド `ip addr` により，経路情報が割り当てられていることを `ip route` により確認する．また，その結果から，割り当てられている IP アドレス，そのアドレスが属するネットワークのアドレス，サブネットマスク長，デフォルトゲートウェイが何であるか調べる．この結果は，報告書中作業環境の説明に記載してあることが望ましい．

3.2.5 端末の捜索

サブネット内の他の端末を，自身の属しているネットワークの IP アドレスなどからそれらの IP アドレスを推測して，ping により反応があるかどうか調査し，端末を発見する．

4 課題 1

この課題は，C++ 言語によってプロセス間通信を実現するプログラムを作成し，ネットワークプログラミングの基礎的な事柄に慣れることを目的とし，UDP および TCP を用いた Echo プログラムを作成する．なお，Echo プログラムとは，サーバがクライアントから送信されてきた文字列をそのまま返信するものである．

4.1 課題 1.1: UDP による Echo プログラム

本課題では UDP を用いた Echo プログラムを作成する．参考コード `udp_time_sever.cpp`, `udp_time_client.cpp`, および `ex_gettimeofday.cpp` を改変して取り組むと良い．以下に，サーバ，クライアントが持つべき機能をそれぞれ示す．カッコ内の関数は使うべきソケットライブラリの関数名を示している．

- サーバはソケットの作成，ポートの割り当て（`bind`），を終えた後，クライアントからの文字列送信を待機する（`recvfrom`）．サーバは終わりなく稼働するように，繰り返し文により接続を待機する．文字列を受信した際に，1) 受信した文字列，2) クライアントの IP アドレス，3) ポート番号をターミナルに表示する．
- クライアントは標準入力からユーザの文字列入力を待機し，入力された文字列をサーバに送信する．繰り返し文を用い，何回でも文字列を入力できるようにする．クライアントプログラムを終了するための入力文字列（`q`，`quit` や `exit` 等）を定義し，プログラムを終了できるようにする．また，文字列を送信する直前から，受信した直後までの時間，Round Trip Time (RTT) [11] を計測して画面に表示する．

作成したプログラムを実行し，以下の項目を確認して考察と共に結果を記載せよ．

1. クライアントから文字列を数回送信し，送信した文字列とサーバからエコーされた文字列が一致すること．
2. RTT を計算し，ターミナルに表示できること．
3. ターミナルを 2 つ立ち上げ，クライアントプログラムを 2 つ同時に実行し，サーバが複数のクライアントからの文字列を正しくエコーできること．

結果の記載に際して，CLI 上に表示された文字列を図として扱い，報告書中で図番号を引用すること．

ヒント

- 補助資料 `sup.pdf` にもソケットライブラリの使い方の概略が載っているので参考にせよ．
- クライアントの IP アドレスとポートは，`struct sockaddr_in` 構造体の `sin_addr` と `sin_port` から取得する．また，IP アドレス（`sockaddr_in` の `sin_addr`）を文字列に変換するには，`arp/inet.h`（Unix 系の場合）で定義されている関数 `inet_ntoa()` を用いる．
- 文字化けが起こる場合は，関数 `recvfrom` の戻り値が受信した文字列長さであるため，この文字列長さを用いて配列の文字列終了位置と思われる場所に終端記号 `\0` を入れることで対処する．
- RTT の計算は `ex_gettimeofday.cpp` を参考にする．

- 127.x.x.x のようなループバックアドレスを試験に用いると RTT の違いが出づらいため，可能なら他人の端末へ接続し確認する．ループバッグアドレスを用いて試験を行った場合はその点を考慮した考察を書く．
- Failed to bind a socket とエラーが出たら，しばらく待つ．このエラーは以前使っていた TCP の接続に対する確認応答パケットが送られてくる可能性があり，それを新しい TCP 接続で受け取ることを防ぐために，OS が新規 TCP 接続の立ち上げを禁止している期間に起こる．クライアント側から接続を切ればこの現象は起こりづらい．
  - macOS の場合は 5000 番ポートがコントロールセンターに使用されている場合がある．5001 など他の値に設定するとよい．
  - それでも解決しなければ lsof -P -i:**<ポート番号>**などでそのポートが使用されていないか確認せよ．

4.2 課題 1.2: TCP による Echo プログラム

課題 1.1 で作成したサーバ，クライアントプログラムと同等の機能を持つものを作成せよ．TCP を用いた場合と UDP を用いた場合で RTT や送信結果に違いがでるか確認せよ．ただし，課題 1.1 で要求されていた同時接続の試験に関して，TCP ではあるクライアントが通信を行っている間新しい接続要求は待ち行列に格納され待機状態となるため，通信を複数クライアントで同時に行うことは，サーバの処理を並列化しなければ行うことはできない．サーバの並列化は難しいため，課題 1.2 では接続待ちとなることを確認するだけに留めるものとする．本課題は tcp_time_sever.cpp，tcp_time_client.cpp，および ex_gettimeofday.cpp を改変して取り組むと良い．

ヒント

- UDP と違い，TCP を用いる場合は通信の開始前に接続を確立する必要があり，このための手順が増える．具体的には，サーバ側では bind によりポートを割り当てた後，listen により待ち行列を生成する．この待ち行列は OS が管理するものであり，到着した接続要求が格納されているため，accept により接続要求を取り出し，通信を開始する．クライアント側でも，データ送受信と接続を同時に行っていた UDP の場合と異なり，connect で接続要求を出してから read, write で通信を行う．
- read はデータが届くまでプログラムをその場所で休止させる．何かしらデータが届いていればそれを取り出すが，届いている分だけ先に渡すため，クライアントが write で書き込んだ分全てを一度に受信できるとは限らない．1 パケットに収まる範囲ならだいたい一度に取り出せる．

4.3 課題 1.3: TCP・UDP による送信時間

課題 1.1，および課題 1.2 で作成したプログラムを拡張し，文字列の長さを，様々に変化させて，RTT の違いを観測し，グラフで結果を示す．$1, a, ..., a^n$ のように長さを変化させると良い．$a = 1.2$ など実験時間の長さと結果の正確性を考慮し調整して使う．実験は複数回実施し平均の結果を用いると考察をよく行うことができる．70000 以上のデータ長さの場合まで実験を行うと良い．長い文字列は，hint-network.txt に記載の方法で生成すると良い．TCP，UDP における結果の違いを確認し，考察する．

ヒント

- データの最後に，終端文字を配置することで，データの終了を受信側に伝える．
- 文字列が一致していることをすべての文字を調べて確認すること．UDP では欠損が起こる可能性がある．
- hint-network.txt も確認すること．

5 課題 2

本課題では TCP を用いて簡易掲示板を作成する．アプリケーションはクライアント・サーバモデルで作成する．興味と自信が有るなら Peer-to-peer (P2P) で作っても良いが（ただし，P2P の場合は実験室でないと難しいかも知れない)，ここではサーバ・クライアントモデルで作成するものとしてプログラムの仕様を説明する．また，ここでの記載通りに全員が作っていれば他の班とも雑談が行えるだろう．

クライアントはサーバに宛先を記した伝言を送る．これはつまり小さなメールと同等のものである．サーバはこれら伝言を保管しておき，クライアントからの要求に応じて伝言を送信するものとする．このために，クライアントが初めに送るデータには，伝言の送信か，取得かを区別し，また宛先も区別可能な情報が必要である．また，文字列を反復するだけのアプリケーションにくらべこの仕組みは複雑であることから誤動作が起こりやすいため，送られてきたデータが本アプリケーションのものであるかどうかを識別しておいたほうが良い．

その他，ネットワークアプリケーションでは，複数の Byte にまたがる数値を扱う場合エンディアン [12] という Byte 順序について気にしなければならない．ただし，これとポインタ操作を合わせて扱うことは初学者には難しいため，本課題ではすべて可読な文字列に符号化して送信することとする．各要素は半角空白か改行コードで区別し，文章など空白を含む文字列を送りたい場合は `\"` で囲むものとする．このような符号化方式で命令を送り，サーバ側に演算を実行させる例として，`command.client.cpp` と `command.server.cpp` を用意したため，参考にすると良い．

以上の仕様に対し，クライアントは次のように処理を行う．

1. ユーザに送信か受信かを指定するように画面へ指示を表示したのち，ユーザの入力を標準入力より読み取る．
2. ユーザ名を入力するよう画面に表示したのち，ユーザの入力を標準入力から読み取る．
3. ユーザに宛名を入力するよう画面に表示したのち，ユーザの入力を標準入力から読み取る．
4. 送信の場合のみ: ユーザに伝言を入力するよう画面に通知し，ユーザの入力を標準入力から読み取る．
5. 上記宛名と伝言を用い，サーバに処理を依頼する．
6. サーバからの応答を画面に表示する．

クライアントが送るデータは次のように定める．

1. 初めの文字列: 乱数による署名 `1iSAUYKj` を記載．これを初めに確認することで他のアプリケーションプロトコルの受信を防ぐ．
2. 以降の文字列: 以下の命令 `get-from`，`send-to` のいずれかを記載する．各要素が空白を含む場合は `\"` で囲む．
3. 取得命令の場合: 文字列 `get-from name1 name2` を送信する．ただし `name1` は受信者の名前，`name2` は送信者の名前で置き換える．`name2` に `all` と記載した場合は全員に対し送られたものを取得することを意味するものとする．
4. 送信命令: 文字列 `send-to name1 name2 massage` を書き込む．ただし `name1` は送信者の名前，`name2` は受信者の名前で置き換え，`name2` に `all` と記載した場合全員に対し送信することを意味するものとする．`massage` には伝言を記載する．

サーバはクライアントから接続要求を待機し，上記内容が送られてくることを想定して `read` によりデータを読み込む．送られてきたデータに対し，次の通り処理を行う．

1. 初めの文字列が，署名 `1iSAUYKj` と一致しない場合は接続を `close` により通信を遮断する．
2. ２つめの文字列が `get-from` の場合（取得命令）: `name1` が宛先，`name2` が送信者である伝言をすべて返答したのち通信を遮断する．各伝言は今までの命令と同様半角空白と `\"` を合わせて区別するとよい．
3. 送信命令の場合: `name1` を送信者，`name2` を受信者として，`'` **送信元名** `' > message` と連結した文字列とともに保存する．構造体およびクラスで，これら情報を格納可能な型を作り，それを配列で管理すると良い．
4. 命令が上記以外の場合: 処理が失敗したことをクライアントに通知して通信を切断する．

通信を遮断することは，ソケット記述子を close() により破棄することで行う．伝言の最大長さは実用上は 100 万字などに適宜決めておく．最大長さを超えた文字列が送られてきた場合はクライアントに送信の失敗を通知して通信を遮断することがプログラムの動作上安全である．

以上のデータを送り終えたら接続を close により切断する．サーバ側で保持する伝言の数は固定長にしておいて記憶する場所が足りない場合は古いものを上書きして破棄するようにすると良い．伝言は宛先で検索をしなくてならないので，高速性を求める人はデータ構造とアルゴリズムで習った 2 分木を使い実装してみてみ良いだろうが，この程度の課題では線形検索でほとんど問題なく動作する．

ヒント

- 宛名や伝言などはサーバ側，クライアント側で逐一画面に表示して動作を慎重に確認したほうが良い．IP アドレスも表示しておいたほうが良いだろう．
- 文字列の保存には string クラスを使うと良い．
- 伝言の保存にはリングバッファを使うと良い．
- 受信する伝言を時刻ごとに判別したい人，新着の伝言をプログラムに自動で確認させたい人は追加課題に取り組まれると良い．

追加課題

ここまでの課題で作成した簡易掲示板はとても原始的なものであり色々と不便がある．例えば，サーバは一人の宛名に対し伝言は一つしか保持できないため，読み取る前に新しい物が送られてきた場合に過去のものが破棄されてしまう問題が有る．また，クライアントにしても，伝言の送信と確認を手動で行うため，会話が不便である．そこで本課題ではまず宛名あたりの複数伝言に対応し，さらに伝言送信用クライアントと確認用クライアントの 2 つを実装し，操作性の向上を目指す．具体的には次の通りサーバを拡張する．

1. 時間指定を追加: 取得命令を get-from-time name1 name2 time1 time2 のように拡張する．ただし，time1 と time2 は関数 time（tpc_time_client.cpp を参照）で得られる Unix 時刻とする．サーバ側はこの時間指定に次のように対応する．
   (a) 指定された送信者，受信者と時間の範囲内に該当する伝言を全てクライアントへ送る．
   (b) time2 < time1 の場合，最新の伝言のみをクライアントへ送る．
2. 書き込み命令に対しては，サーバ側で時刻を OS より取得して記録しておくだけで，クライアント側の拡張は必要としない．

確認用クライアントは次のように作成する．

1. sleep を使い，数秒毎に新着伝言がないかサーバへ確認し，更新内容があれば画面へ表示する．
2. 最後に確認した時刻と現在時刻の間の伝言のみ確認する．

伝言送信用クライアントは，送信することに特化して，最初に入力した送信者，受信者の情報を固定として連続して伝言を送信できるようにする．ただし，利用者が入力中にサーバが待機しないよう，利用者が入力を終えてから接続，送信，切断の手順を処理するようにする．サーバは待ち行列の数（標準では 5 つ）までしか接続を保留できないため，できるだけ短い時間で通信を終えるように配慮する（サーバ側を並列化すればこの制限はなくなるが少し難しい）．

6 報告書の提出方法と作成要項

次の指示に従い報告書を提出せよ．

- 報告書の作成には何を用いても構わないが，Portable Document Format (PDF) [8] で提出すること．
- 提出は kibaco の指定した場所へ行うこと．

- 不備があるものについては再提出を課す．メールを確認すること．
- 再提出の要求に応じない場合単位を取得できないことに注意すること．
- 再提出で基準に達しない場合，原則として単位を認定しないため注意すること．

報告書は次の指示に従い構成せよ．後述する体裁と合わせすべて技術報告書作成における世界常識であるため必ず身につけよ．

- 最初のページには実験名，氏名，学修番号，共同実験者，改定日時を記載すること．ただし，改定日時は提出するたびに追記し，以前のものは消さないこと．
- 文献を調査した結果得られた知見には引用を記載すること．
- 各課題の報告は見出しを付けて節としてまとめ，それぞれ分けて記載すること．
  1. 文書の構成は，文書全体 ＞ 章 ＞ 節 ＞ 小節 のような階層構造をとり，それぞれに見出しが付けられる．文書全体の見出しは表題と呼ばれる．この構成は箇条書きとは異なる．見出しが文書構造を示し，箇条書きは要素の階層関係を示すことに使われる．
- 節の見出しの構成は次に従う（本実験の報告書には章は用いない）．指導書の構成とは異なるため注意．
  1. はじめに: 実験の目的の説明を行う．例えば得られる知識がなんの役に立つのか，現実のどこで使われているかなどを語る．実験指導書の「はじめに」を参考にすると良い．
  2. 事前調査内容: 宿題の内容を記載する．最初に，事前調査を行うことの意義と調査内容を簡単に説明する．
  3. 実験内容: 実験の内容を説明する．ここではプログラムの出力など結果は載せない．
  4. 実験環境: ハードウェア，コンパイラ，OS 等の実験環境を説明する．
  5. 実験結果: プログラムの出力などを参考に結果を説明する．図表を使う．ここで実験内容の説明は行わない．
  6. 考察: 得られた結果と予想していた結果との違い，結果に対して考えられる原因などについて説明する．
  7. まとめ: 実験の目的，内容と得られた結果を簡単に説明する．共同実験者への謝辞を記載する．
  8. 付録: ソースコードなど
- 初出の略称は商標（ライブラリ名含む）を除き，正式名称を記載すること．例えば，「Internet Protocol (IP) はインターネットの…」と最初は記載し，以降は単に「IP パケットを…」のように書く．商標などは引用か脚注[*1]を示す．
- 話のまとまりごとに段落を分けること．だたし 1-2 行のみの極端に短い段落とならないように中身のある段落を作ること．
- 図表など補助資料は必ず本文中で引用すること．基本的には，「…の結果を図 x に…を示す.」「図 x より，…であることがわかる.」のように説明を行うこと．執筆者は，読者が本文だけを頼りに報告書を読むことつねに意識しなければならない．したがって，数式・画像・ソースコードなど，通常の文章から外れる補助資料はそれが何であるか本文中で説明し，逐一読者の注意を向けなければならない．本文中に引用のない補助資料は評価の対象外となることに注意すること．
- 各課題の報告では，必ず実験および課題の意図，工夫した点，事実および結果，考察を記載すること．例えば，「この実験では…を確認する．このために…を行った．その結果を図 x に示す.」と説明した後，
  1. 予想と一致する場合: この図より，…であることがわかる．これは想定される通りの結果であり，…であることが確かめられたと言える．
  2. 予想が外れる場合: この図より，…であることがわかる．…であることから…の結果になるだろうと考えたが，結果では…となっている．これは，…が原因であると考える．

  のように記載する．実験は何かしらの意図を持って行い，それには期待する結果があるのだから，何を予想したかを思い出せば，それが外れた場合でも考察が書きやすいはずである．考察のない結果は信頼性に欠けるため，責任を持って読者に結果を伝えることを意識する．論文ではありえないが，本当に何も考察すべきことが

[*1] こういったページの下部にあるものと，本文中の印で構成されるものが脚注である．[accessed in Oct., 2020] https://ja.wikipedia.org/wiki/%E8%84%9A%E6%B3%A8

ない場合であっても「この結果に対し特筆すべきことはない」と書くべきである．

報告書の体裁については次の指示に従い提出すること．

- 表題は中央揃え，見出しは左揃え，本文は両端揃え（最終行は左揃え）とすること．
- 行間はこの資料で使われる程度とし，広すぎないように気をつけること．
- 本文のフォントの大きさを統一すること．見出し・表題・図表など補助資料に使うものも統一すること．
- 本文のフォントは明朝体および Roman 体を使うこと．
- 本文に黒以外の色は使わないこと．
- 数式には数式エディタや LaTeX の数式モードを使うこと．
- 図表とその説明（キャプション）は中央揃えで，ページ上部あるいは下部へ詰めて配置すること．
- 図表の説明は図の下部，表の説明は表の上部に配置すること．
- 図表に無用な外枠は入れないこと．
- 図表の軸や文字色も完全に黒とすること（Excel は初期設定で灰色になっている）．
- ソースコードは 1 ページを超えるような長い場合は付録に掲載すること．
- ソースコードは画像ではなく文字列で記載すること．
- 用いる句読点は統一すること．「、。」と「,.」の併用は認められない．
- 引用符は，“このように” 左と右を区別すること．LaTeX では ``'' と入力する．
- 実験結果の有効数字は揃えること．

参考文献

[1] RFC 1180 – TCP/IP tutorial. https://datatracker.ietf.org/doc/html/rfc1180 (Accessed 2024/12/01)

[2] RFC 768 – User Datagram Protocol. https://datatracker.ietf.org/doc/html/rfc768 (Accessed 2024/12/01)

[3] RFC 793 - Transmission Control Protocol. https://datatracker.ietf.org/doc/html/rfc793 (Accessed 2024/12/01)

[4] ISO/IEC 9899:2024 - Information technology — Programming languages — C. https://www.iso.org/standard/82075.html (Accessed 2024/12/01)

[5] The Standard : Standard C++. https://isocpp.org/std/the-standard (Accessed 2024/12/01)

[6] RFC 1122 - Requirements for Internet Hosts - Communication Layers. https://datatracker.ietf.org/doc/html/rfc1122#page-8 (Accessed 2024/12/01)

[7] Bash Reference Manual. https://www.gnu.org/software/bash/manual/bash.html (Accessed 2024/12/01)

[8] Adobe PDF とは？ Portable Document Format（PDF）| Adobe Acrobat https://www.adobe.com/jp/acrobat/about-adobe-pdf.html (Accessed 2024/12/01)

[9] ISO/IEC 7498-1 The Basic Model, 1994

[10] Time to live, Wikipedia. https://en.wikipedia.org/wiki/Time_to_live (Accessed 2024/12/01)

[11] Round-trip delay, Wikipedia. https://en.wikipedia.org/wiki/Round-trip_delay (Accessed 2024/12/01)

[12] Endianness, Wikipedia. https://en.wikipedia.org/wiki/Endianness (Accessed 2024/12/01)