Ciscoネットワーク演習2　第4週　　 クラス　　　　番号　　　　氏名

• 14 - トランスポート層

14.0 - 概要

14.0.1 - このモジュールを学ぶ理由

トランスポート層は、名前が示すように、データがホスト間で転送される場所です。これはあなたのネットワークが本当に動作する場所です！トランスポート層は、TCPとUDPの2つのプロトコルを使用します。TCPは、郵便で書留を受け取ると考えてください。UDPは、切手を貼った手紙に似ています。このトピックでは、トランスポート層での TCP と UDP の仕組みについて説明します。

14.0.2 - このモジュールで学ぶこと

**データの転送：**トランスポート層の管理における目的を説明してください。 エンドツーエンドの通信におけるデータの転送

**TCP の概要：**TCPの特徴を説明

**UDP の概要：**UDPの特性を説明。

**ポート番号：**TCPとUDPがどのようにポート番号を使用するかを説明

**TCP 通信プロセス：**TCP セッションの確立と終了のプロセスを説明 信頼性の高い通信を促進します。

**信頼性とフロー制御：**TCP プロトコルのデータが配信を保証するために、どのように送信され、どのように確認されるかを説明

**UDP通信：**エンドツーエンド通信のサポートにおける、トランスポート層プロトコルの動作を比較

14.1 - データの転送

14.1.1 - トランスポート層の役割

アプリケーション層プログラムは、送信元ホストと宛先ホスト間で交換する必要があるデータを生成します。トランスポート層は、異なるホスト上で実行されているアプリケーション間の論理的な通信を担当します。これには、2つのホスト間の一時的な**[** ① **セッション ]**の確立や、アプリケーションに対する信頼性の高い情報伝送などのサービスが含まれます。

14.1.2 - トランスポート層の責任

**個々の通信の追跡：**トランスポート層では、ソースアプリケーションとデスティネーションアプリケーションの間を流れるデータの各セットは通信として知られており、個別に追跡されます。

**データのセグメント化とセグメントの再構築：**アプリケーションデータを適切なサイズのブロックに分割するのはトランスポート層の責任です。

**ヘッダー情報の追加：**トランスポート層プロトコルは、データの各ブロックに、いくつかのフィールドに編成されたバイナリデータを含むヘッダー情報を追加する。

**アプリケーションの識別：**トランスポート層はポート番号と呼ばれる識別子を使用してターゲットアプリケーションを識別します。

**通信の多重化：**トランスポート層は、セグメンテーションと多重化を使用して、異なる通信を同じネットワーク上で交互に行えるようにする。

14.1.3 - トランスポート層プロトコル

IP は、パケットの構造、アドレッシング、およびルーティングだけに関係します。IP は、パケットの配信または転送の方法を指定しません。

トランスポート層のプロトコルは、ホスト間でメッセージを転送する方法を指定し、通信の信頼性要件を管理する責任があります。トランスポート層にはTCPとUDPのプロトコルがあります。

14.1.4 - 伝送制御プロトコル（TCP）

TCPは信頼性が高く、高機能なトランスポート層プロトコルです。すべてのデータが確実に宛先に到着するようになっています。

通信の状態を維持し、情報を追跡するために、TCPはまず、送信者と受信者の間の接続を確立する必要があります。これが、TCPがコネクション型プロトコルとして知られている理由です。

14.1.5 - ユーザ データグラム プロトコル（UDP）

UDP はコネクションレス型プロトコルである。UDPは信頼性とフロー制御を提供しないため、確立された接続は必要ありません。

UDP は、宛先でデータが受信されたことを示す確認応答がないため、**[** ② **ベストエフォート配信プロトコル ]**とも呼ばれます。UDP では、配信が成功したことを送信側に知らせるトランスポート層プロセスがありません。

14.1.6 - アプリケーションに適したトランスポート層プロトコル

一部のアプリケーションでは、ネットワーク上での伝送中に多少のデータ損失は許容できますが、遅延は許容できません。これらのアプリケーションでは、ネットワークのオーバーヘッドが少なくて済むため、**[ ③ UDP ]**の方が適しています。

14.1.7 - 理解の確認-データの輸送

14.2 - TCP の概要

14.2.1 - TCPの機能

このトピックでは、TCPが何をするのか、UDPではなくTCPを使用するのが良いと思われる場合について詳しく説明します。

TCPは、データのセグメンテーションと再構築の基本機能をサポートするだけでなく、以下のサービスも提供しています。

**セッションの確立：** TCP は、トラフィックを転送する前に、送信元デバイスと宛先デバイス間の永続的な接続（またはセッション）をネゴシエートして確立するコネクション指向プロトコルです。セ

**信頼性の高い配信の保証：** TCP は、送信元から送信された各セグメントが宛先に到着することを保証します。

**同一順序配信を提供：**セグメントの番号付けと順序付けにより、TCP はセグメントが適切な順序に再組み立てられるようにします。

**フロー制御のサポート：** TCP は、これらのリソースが酷使されていることを認識している場合、送信側アプリケーションにデータフローの速度を下げるように要求できます。これは、送信元が送信するデータの量を調整する TCP によって行われます。

14.2.2 - TCP ヘッダー

TCPは、通信セッションの状況を追跡するステートフルプロトコルです。セッションの状態を追跡するために、TCP はどの情報を送信したかと、どの情報が確認応答されたかを記録します。ステートフル セッション は、 セッションの 確立から始まり、 セッションの 終了とともに終了します。

タイムライン

自動的に生成された説明TCPセグメントは、アプリケーション層データをカプセル化するときに、20バイト（160ビット）のオーバーヘッドを追加します。

14.2.3 - TCP ヘッダーフィールド

この表は、TCP ヘッダーの **[** ④ **10 ]**個のフィールドを説明しています。

|  |  |
| --- | --- |
| **TCP ヘッダーフィールド** | **説明** |
| 送信元ポート | 送信元アプリケーションをポート番号で識別するために使用する 16 ビットフィールド。 |
| 宛先ポート | 宛先アプリケーションをポートによって識別するために使用される16ビットのフィールド 16 ビットフィールド。 |
| シーケンス番号 | 送信するデータに、順序を付けるため使用される32ビットのフィールド。 |
| 確認応答番号 | データが受信されたことを示し、次に送信元に求めるバイトを示す32ビットのフィールド。 |
| ヘッダー長 | 「データオフセット」とも呼ばれ、TCPセグメントヘッダーの長さを示す4ビットのフィールド。 |
| 予約済み | このフィールドは、将来的な使用のために予約されています。 |
| 制御ビット（6） | TCPセグメントの目的と機能を示すビットコードまたはフラグを含む、 使用される6ビットのフィールド。 |
| ウィンドウサイズ | 宛先アプリケーションをポート番号によって識別するために使用される16ビットのフィールド。 |
| チェックサム | セグメントヘッダーとデータのエラーチェックに使用される16ビットのフィールド。 |
| 緊急 | 含まれるデータが緊急であるかどうかを示すために使用される16ビットのフィールド。 |

14.2.4 - TCP を使用するアプリケーション

TCPは、データストリームをセグメントに分割することに関連するすべてのタスクを処理し、信頼性を提供し、データフローを制御し、セグメントを並べ替えます。

ダイアグラム

自動的に生成された説明

14.2.5 - 理解度チェック - TCPの概要

14.3 - UDP の概要

14.3.1 - UDPの機能

UDP はベストエフォート型の配信プロトコルです。UDPは、TCPと同じデータのセグメンテーションと再構成を提供する軽量なトランスポートプロトコルですが、TCPの**[** ⑤ **信頼性とフロー制御 ]**はありません。UDPの機能としては、以下のようなものがあります。

* データは、受信された順序で再構築されます。
* 失われたセグメントは再送信されません。
* セッションの確立は行いません。
* 送信側にリソース利用の可能性は、知らされません。

14.3.2 - UDP ヘッダー

UDPはステートレスプロトコルであり、クライアントもサーバも通信セッションの状態を確認しません。トランスポート層プロトコルとして UDP を使用するときに信頼性が必要な場合は、アプリケーションで処理する必要があります。

グラフィカル ユーザー インターフェイス が含まれている画像

自動的に生成された説明

14.3.3 - UDP ヘッダーフィールド

この表は、UDP ヘッダーの **[** ⑥ **４つ ]**のフィールドを説明しています。

|  |  |
| --- | --- |
| **UDP ヘッダーフィールド** | **説明** |
| 送信元ポート | 送信元アプリケーションをポート番号で識別するために使用する 16 ビットフィールド。 |
| 宛先ポート | 宛先アプリケーションをポート番号によって識別するために使用される16ビットのフィールド。 |
| 長さ | ヘッダとデータを含むUDPデータグラム全体のバイト数を示す16ビットフィールド。 |
| チェックサム | データグラムヘッダーとデータのエラーチェックに使用される 16 ビットフィールド。 |

14.3.4 - UDP を使用するアプリケーション

UDP に最も適したアプリケーションには、次の 3 種類があります。

**ライブビデオおよびマルチメディアアプリケーション:**これらのアプリケーションは、いくつかのデータ損失を許容することができますが、ほとんどまたはまったく遅延を必要としません。たとえば、VoIP やライブストリーミングビデオなどがあります。

**単純な要求および応答アプリケーション:**ホストが要求を送信し、応答を受信する場合と受信しない場合があり、単純なトランザクションを持つアプリケーション。例としては、DNS や DHCP などがあります。

**自身で信頼性を扱うアプリケーション:**フロー制御、エラー検出、確認応答、エラー回復が不要な方方向通信であるか、アプリケーションが扱う。たとえば、SNMP や TFTP などがあります。

ダイアグラム

自動的に生成された説明

14.3.5 - 理解度チェック - UDPの概要

14.4 - ポート番号

14.4.1 - 複数の個別通信

TCP および UDP トランスポート層プロトコルは、ポート番号を使用して、複数の同時通信を管理します。

送信元ポート番号はローカルホスト上の送信元アプリケーションに関連付けられ、宛先ポート番号はリモートホスト上の宛先アプリケーションに関連付けられます。

14.4.2 - ソケット ペア

送信元ポートと宛先ポートはセグメント内に付加されます。その後、セグメントは IP パケット内にカプセル化されます。IP パケットには送信元と宛先の IP アドレスが含まれています。送信元 IP アドレスと送信元ポート番号、または宛先 IP アドレスと宛先ポート番号を組み合わせたものを**[** ⑥ **ソケット ]**といいます。

14.4.3 - ポート番号グループ

送信元と送信先のポート番号を識別するために使用される16ビットは、0から65535までのポートの範囲を提供します

|  |  |
| --- | --- |
| **ポートグループ** | **番号の範囲** |
| 既知の（well-known）ポート | 0～1,023 |
| 登録済ポート | 1,024～49151 |
| プライベート および/ または ダイナミックポート | 49152 ～ 65535 |

既知の（well-known）ポート番号とそれに関連するアプリケーション

|  |  |  |
| --- | --- | --- |
| ポート番号 | プロトコル | アプリケーション |
| 20 | TCP | ファイル転送プロトコル（FTP） |
| 21 | TCP | ファイル転送プロトコル（FTP） - 制御 |
| 22 | TCP | Secure Shell (SSH) |
| 23 | TCP | Telnet |
| 25 | TCP | Simple Mail Transfer Protocol（SMTP） |
| 53 | UDP/TCP | Domain Name Service（DNS） |
| 67 | UDP | Dynamic Host Configuration Protocol (DHCP) - Server |
| 68 | UDP | Dynamic Host Configuration Protocol - Client |
| 69 | UDP | Trivial File Transfer Protocol (TFTP) |
| 80 | TCP | Hypertext Transfer Protocol（HTTP） |
| 110 | TCP | Post Office Protocol version 3（POP3） |
| 143 | TCP | IMAP（Internet Message Access Protocol） |
| 161 | UDP | Simple Network Management Protocol（SNMP） |
| 443 | TCP | Hypertext Transfer Protocol Secure (HTTPS) |

14.4.4 - netstat コマンド

誰かがローカルホストに接続していることを示しています。ネットワーク接続されたホスト上で、どのアクティブなTCP接続が開いていて実行されているかを知る必要がある場合があります。Netstat は、これらの接続を確認するために使用できる重要なネットワークユーティリティです。

14.4.5 - 理解度チェック - ポート番号

14.5 - TCP 通信プロセス

14.5.1 - TCP サーバプロセス

このトピックでは、TCP スリーウェイハンドシェイクおよびセッション終了プロセスについても学習します。

プロセスの詳細はWebテキスト参照

14.5.2 - TCPコネクション確立

TCP 接続では、ホストクライアントは **[** ① **3 ウェイハンドシェイクプロセス ]** を使用してサーバとの接続を確立します。

ステップ1. **SYN**：開始側クライアントは、サーバとのクライアント-サーバ間セッションを要求します。

ステップ2. **ACKとSYN：**サーバがクライアントからサーバへの通信セッションを受け入れ、サーバからクライアントへの通信セッションを要求します。

ステップ3. **ACK：**開始側クライアントがサーバからクライアントへの通信セッションを受け入れます。

14.5.3 - セッションの終了

TCP でサポートされる単一のメッセージ交換を終了するためには、**[** ② **4 回 ]**の交換によって両方のセッションを終了する必要があります。

ステップ1. **FIN：**ストリームで送信するデータがクライアント側にもうなければ、クライアントが FIN フラグをセットしたセグメントを送信します。

ステップ2. **ACK：**サーバが ACK を送信して、クライアントからサーバへのセッションを終了するための FIN を受信したことをクライアントに知らせます。

ステップ3. **FIN：**サーバがサーバからクライアントへのセッションを終了するための FIN をクライアントに送信します。

ステップ4. **ACK：**クライアントが ACK で応答して、サーバからの FIN を受け入れます。

14.5.4 - TCPスリーウェイハンドシェイクの分析

TCPは全二重プロトコルで、各接続は2つの一方通行の通信セッションを表します。接続を確立するために、ホストは 3 ウェイハンドシェイクを実行します。

通信が完了するとセッションが終了し、接続が終了します。接続とセッションのメカニズムによって、TCP の信頼性機能が実現されます。

TCP セグメント ヘッダーの制御ビットの**[** ③ **6 ]**ビットは、フラグとも呼ばれます。フラグは、「on」または「off」に設定されているビットです。

14.5.5 - ビデオ：TCP 3 ウェイ ハンドシェイク

ビデオ参照

14.5.6 - 理解度チェック - TCP通信プロセス

14.6 - 信頼性とフロー制御

14.6.1 - TCPの信頼性-配信の保証

TCP セグメントが宛先に到達しない場合があります。また、TCP セグメントが順番通りに到着しないこともあります。受信者が元のメッセージを理解するには、すべてのデータを受信し、これらのセグメント内のデータを元の順序に再構成する必要があります。この目標を達成するために、各パケットのヘッダーに**[** ④ **シーケンス番号 ]**が割り当てられます。シーケンス番号は、TCP セグメントの最初のデータ バイトを表します。

14.6.2 - ビデオ - TCP の信頼性 - シーケンス番号と ACK 応答

TCP の機能の 1 つは、各セグメントが宛先に到達することを保証することです。宛先ホスト上の TCP サービスは、宛先アプリケーションによって受信されたデータを確認します。

ビデオ参照

14.6.3 - TCP の信頼性 - データ損失と再送信

ネットワークがどれほど適切に設計されていても、データ損失が発生することがあります。TCP は、これらのセグメント損失を管理する方法を提供します。これらの中には、確認応答されていないデータのためにセグメントを再送するメカニズムがあります。

シーケンス（SEQ）番号と確認応答番号（ACK）はともに、送信セグメントに含まれるデータのバイトの受信を確認するのに使われます。SEQ番号は送信されるセグメントに含まれるデータの最初のバイトを示します。TCPは、ACK番号を送信元に送り返すことで、受信者が次に受信することを期待しているバイトを送信者に示します。これを期待確認応答と呼びます。

14.6.4 - ビデオ - TCP の信頼性 - データの損失と再送信

ビデオ参照

14.6.5 - TCP フロー制御-ウィンドウサイズと確認応答

TCP はフロー制御のメカニズムを備えています。フロー制御とは、送信先が受信して確実に処理できるデータ量のことです。フロー制御は、ある特定のセッションについて送信元と宛先の間のデータ フローの速度を調整することで、TCP 伝送の信頼性を維持するのに役立ちます。そのために、TCP ヘッダーには[ ⑤ **ウィンドウ サイズ** ]と呼ばれる 16 ビット フィールドが含まれています。

ウィンドウサイズは、確認応答を受信する前に送信できるバイト数を決定します。確認応答番号は、次に期待されるバイトの数です。

14.6.6 - TCP フロー制御-最大セグメントサイズ (MSS)

MSS は、デバイスが 1 つの TCP セグメントで[ ⑥ **受信できるデータの最大量** ] (バイト単位) を指定する TCP ヘッダーのオプションフィールドの一部です。MSS サイズには TCP ヘッダーは含まれません。MSS は通常、3 ウェイハンドシェイク中に含まれます。

IPv4 を使用する場合、一般的な MSS は 1,460 バイトです。

14.6.7 - TCPフローコントロール - 輻輳回避

ネットワーク上で輻輳が発生すると、過負荷のルータによってパケットが破棄されます。

輻輳が発生すると、失われた TCP セグメントが送信元から再送信されます。再送信が適切に制御されていない場合、TCP セグメントの追加再送信によって、輻輳がさらに悪化する可能性があります。  
TCP セグメントがタイムリーに確認応答されていないか、確認応答がないと送信元が判断した場合、[ ⑦ **確認応答を受信する前** ]に送信するバイト数を減らすことができます

送信元が送信する未認識バイト数を減らしているのであって、送信先によって決定されるウィンドウサイズを減らしているのではないことに注意してください。

14.6.8 - 理解の確認-信頼性とフロー制御

14.7 - UDP通信

14.7.1 - UDPの低オーバーヘッドと信頼性

前に説明した通り、UDPはVoIPのように、高速である必要がある通信に最適です。このトピックでは、UDP が一部の種類の送信に最適である理由について詳しく説明します。UDP は接続を確立しません。UDP は、データグラムヘッダーが小さく、ネットワーク管理トラフィックがないため、[ ⑧ **オーバーヘッドの少ない** ]データトランスポートを提供します。

14.7.2 - UDP データグラムの再構成

CP のセグメントと同じく、宛先に UDP データグラムが送信されると、それらのデータグラムは異なるパスを通ることによって間違った順序で到達することがよくあります。UDP は、TCP のようにシーケンス番号を追跡することはありません。UDPでは、データグラムを送信順に並べ替える方法がありません。

そのため、UDPは[ ⑨ **受信した順** ]にデータを再構築してアプリケーションに転送するだけです。

14.7.3 - UDP サーバーのプロセスと要求

TCP ベースのアプリケーションと同様、UDP ベースのサーバ アプリケーションにも既知の（well-known）ポート番号または登録済みポート番号が割り当てられます。これらのアプリケーションまたはプロセスがサーバー上で実行されている場合、割り当てられたポート番号と一致するデータを受け入れます。UDP は、これらのポートの 1 つ宛てのデータグラムを受信すると、そのポート番号に基づいてアプリケーションデータを適切なアプリケーションに転送します。

14.7.4 - UDP クライアントプロセス

TCP の場合と同様、クライアント/サーバ通信は、サーバ プロセスにデータを要求するクライアント アプリケーションによって開始されます。UDP クライアント プロセスは、ポート番号の範囲から[ ⑩ **動的** ]にポート番号を選択し、それをメッセージ交換のための送信元ポートとして使用します。通常、宛先ポートはサーバ プロセスに割り当てられた既知の（well-known）ポート番号または登録済みポート番号です。

クライアントが送信元ポートと宛先ポートを選択すると、トランザクション内のすべてのデータグラムのヘッダーで同じポートのペアが使用されます。サーバからクライアントに返されるデータについては、データグラムヘッダーの送信元ポートと宛先ポート番号が逆になります。

14.7.5 - 理解度の確認-UDP通信

14.8 - モジュール演習とクイズ

14.8.1 - Packet Tracer -TCP および UDP 通信

14.8.2 - このモジュールで学んだこと

14.8.3 - モジュールクイズ-トランスポート層