PR Rechnernetze und Netzwerkprogrammierung 621.801/621.802	WS 2023/2024
Institut für Informationstechnologie (ITEC)	Rainer/Timmerer
Übungsblatt 05	

Ü 5.1 Das UDP-Protokoll

Wenden Sie Ihr Wissen aus der Vorlesung an und sehen Sie sich zusätzlich RFC 768 (User Datagram Protocol - UDP) an und beantworten Sie folgende Fragen:

- Welche Funktionalität stellt UDP **(nicht)** zur Verfügung?
- Wie sehen die UDP-Header aus?
- Was ist ein Pseudo-Header?
- Wie groß können UDP-Segmente sinnvollerweise sein?

Geben sie Beispiele an, wo UDP verwendet wird (wurde).

Ü 5.2 aauTCP/IP-Referenzmodell: Transportschicht

Nachdem Sie im letzten Übungsblatt ein Request-Response-Protokoll (**vereinfachtes HTTP, erweitern Sie das Request-Response mittels den aauHTTP Stubs!)** zwischen einem Client und einem Server auf der Applikationsebene realisiert haben, sollen Sie nun die Datenübertragung auf der Transportebene realisieren. Als Protokoll für den Transport soll (der Einfachheit halber) UDP verwendet werden.

Achtung: HTTP wird eigentlich über TCP realisiert; dies wird in den nachfolgenden Aufgaben implementieren!

Dieses Protokoll dient nur zum Anwendungs-(De)Multiplexing bzw. zum Ver-/Entpacken der HTTP-Nachrichten (`HTTPClientMsg` und `HTTPServerMsg`).

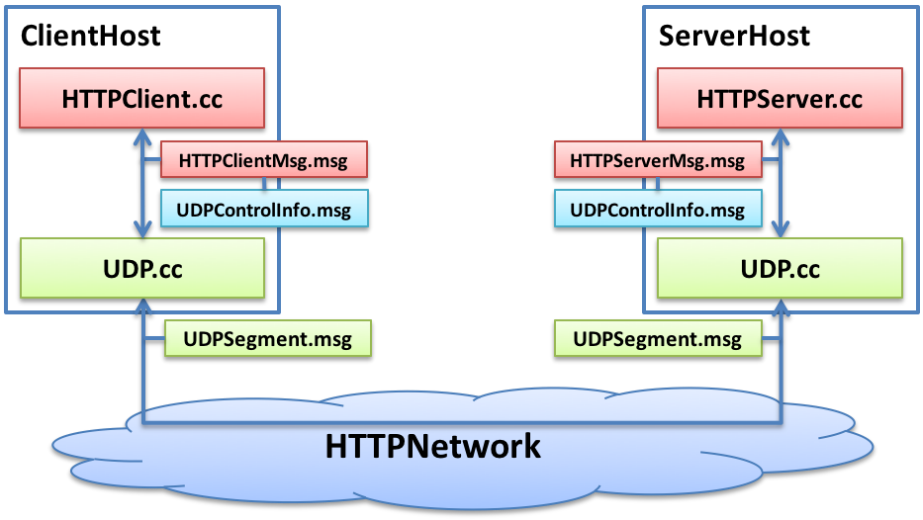


Abbildung 1. UDP-basierte HTTP Kommunikation auf der Transportschicht.

Jede HTTP-Nachricht sollte in einem UDP-Segment versendet werden. Verwenden Sie zum Implementieren der `UDPSegement.msg` (der Einfachheit halber) nur Quell- und Zielports (d.h., `int srcPort` und `int destPort`). Nutzen Sie zum Verpacken der HTTP-Nachrichten die Methode `encapsulate()`, zum Auspacken die Methode `decapsulate()` **. Dazu müssen alle Nachrichten vom Typ `packet` sein.**

Neben den HTTP-Nachrichten müssen auch zusätzliche Informationen zur Transportschicht übergeben werden. Dies geschieht mit der OMNeT++-Message `UDPControlInfo.msg` bestehend aus `InetAddress srcIP`, `int srcPort`, `InetAddress destIP` und `int destPort`. **Achtung**:

Die `UDPControlInfo.msg` ist nicht teil des UDP-Segments und muss vor dem Versenden entfernt werden bzw. an der Empfängerseite wieder erzeugt werden. **Hinweis:** Die `srcIP` bzw. `destIP` wird erst in den kommenden Übungsblättern (Netzwerkschicht) relevant und kann vereinfacht gehandhabt werden (z.B. 127.0.0.1 bzw. vordefinierte Werte).

Erweitern Sie die vorgegebene Datei, um alle notwendigen Informationen übergeben zu können und binden Sie diese in die Kommunikation zwischen den Schichten in den Dateien der Applikationsschicht ein (`HTTPServer.cc` und `HTTPClient.cc`).

***app*-Verzeichnis:**

- Wie im Übungsblatt 4
- Zur Implementierung der Funktionalität des Clients bzw. des Servers verwenden Sie bzw. erweitern Sie die Implementierung von Übungsblatt 4. D.h. beim Senden erzeugen Sie die `UDPControlInfo` und fügen Sie diese an die `HTTPClientMsg` bzw. `HTTPServerMsg` (mit `..->setControlInfo()`) an bzw. beim Empfangen können Sie diese wiederum entfernen (mit `..->removeControlInfo()`).

***udp*-Verzeichnis:**

- **UDP.ned**: Eine einfache Definition eines UDP-Moduls. Hier müssen Sie nichts ändern.
- **UDPSegment.msg**: Die Definition des UDP-Segments.
- **UDPControlInfo.msg**: Die Definition der UDP-ControlInfo. Hier müssen Sie nichts ändern.
- **UDP.cc** bzw. **UDP.h**: Die Implementierung UDP-Schicht. Achten Sie auf eine korrekte Unterscheidung zwischen `arrivedOn("fromUpperLayer")` und `arrivedOn("fromLowerLayer")` sowie darauf, dass die `UDPControlInfo` nicht über das Netzwerk verschickt wird, sondern nur zur Kommunikation zwischen den Schichten verwendet wird (d.h. vertikal, nicht horizontal).

***networks*-Verzeichnis:**

- **Client/ServerHost.ned**: Eine Definition des HTTP-Client/Server-Knotens. Definieren Sie diesen anhand von Abbildung 1.
- **HTTPNetwork.ned**: Die Definition des Netzwerks, das simuliert werden soll. Verwenden Sie die Netzwerkdefinition von Übungsblatt 4 und verbinden Sie ClientHost mit ServerHost.

Ü 5.3 Das TCP-Protokoll

Wenden Sie Ihr Wissen aus der Vorlesung an und beantworten Sie folgende Fragen:

1) Wie wird eine Verbindung beim TCP-Protokoll aufgebaut?
2) Wie wird eine Verbindung beim TCP-Protokoll abgebaut?
3) Können bei der Übertragung durch das TCP-Protokoll TCP-Segmente verloren gehen? Wenn ja: Was passiert dann?
4) Wie werden die Sequenznummern lt. RFC vergeben?

Ü 5.4 Der vereinfachte TCP-Verbindungsaufbau

Nachdem Sie im letzten Übungsblatt UDP als Transportprotokoll eingesetzt haben, sollen Sie nun die Datenübertragung für die Applikation auf der Basis von TCP realisieren.

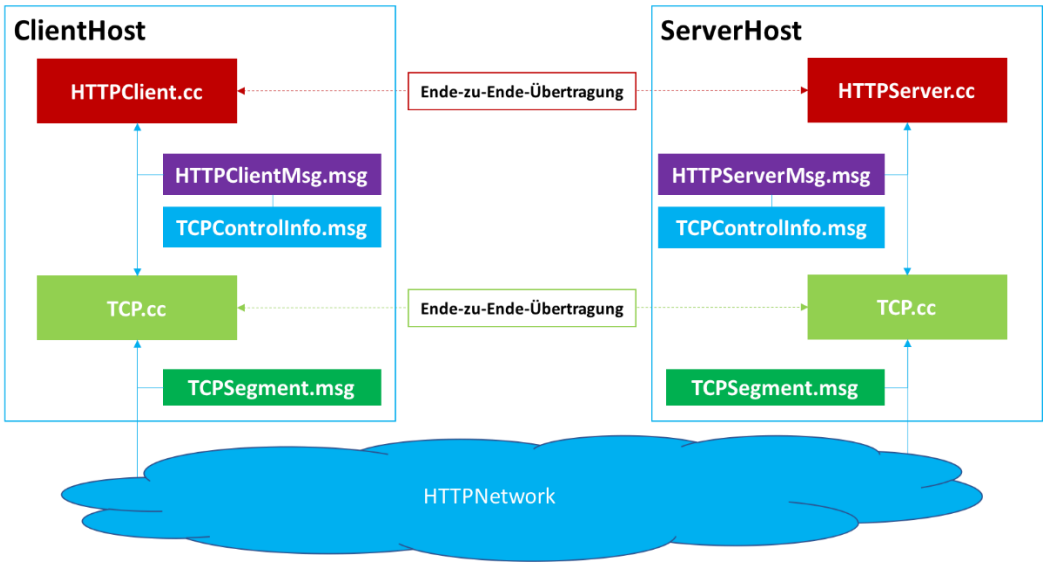
Hinweis: Wie müssten NED-Dateien und Ihre Implementierung aussehen, wenn sowohl UDP als auch TCP gleichzeitig unterstützt werden? Skizzieren Sie die mögliche Lösung und, wenn mögliche, implementieren Sie sie auch.

Achten Sie dabei auch auf den richtigen Einsatz der Nummern in den SEQ- und ACK-Feldern der TCP-Header.

Hinweis: Starten Sie der Einfachheit halber mit einer beliebigen SEQ-Nummer (z.B.: 100 oder 300).

TCP ist verbindungsorientiert und liefert in der Spezifikation genaue Angaben für den Verbindungsaufbau mit. Beschränken sie sich dabei auf den vereinfachten **Verbindungsaufbau** (vgl. Figure 7 von RFC793) und erstellen Sie **zuvor** ein Sequenzdiagramm, das den Verbindungsaufbau darstellt (z.B. mit Hilfe von https://www.websequencediagrams.com/).

Nutzen Sie die TCPControlInfo u.a. zur Steuerung der TCP-Verbindung (Verbindungsaufbau, Verbindungsabbau).



Hinweis: Im nächsten Übungsblatt implementieren wir eine vereinfachte Datenübertragung auf Basis von stop-and-wait (inkl. persistente TCP-Verbindung) sowie den TCP-Verbindungsabbau.