Programmieraufgabe als APL – zu bearbeiten in Gruppen von 4 -5 Mann

Realisierung eines zuverlässigen Multicast-Protokolls mittels Negativer Acknowledgements auf der Basis von UDPv6

Schreiben Sie ein Sender- und ein Empfänger-Programm in C (Konsolenprogramm), mit Hilfe derer eine Textdatei zeilenweise via UDPv6/IPv6- übertragen werden kann. Dem Sender und den Empfängern werden der Dateiname (zu sendende Datei bzw. Dateiname für die Speicherung), die IPv6- Multicast-Adresse und der Port als *Programmargumente* übergeben, um eine Datei *zeilenweise* auf der Basis eines ARQ-Protokollmechanismus und einer vorgebbaren Fenstergröße von 1-10 Paketen (die Fenstergröße ist ebenfalls als Argument zu übergeben, falls diese größer 1 sein soll) versendet werden kann.
Die Multicast-Empfänger sollen anhand von Sequenznummern in den Paketen die Sendereihenfolge der übertragenen Datagramme überprüfen und inkorrekt empfangene Pakete per Unicast negativ bestätigen (NACK). Bei Eintreffen einer NACK - Quittung wird das angeforderte Paket nach dem *Selective Repeat* - Prinzip vom Sender erneut übermittelt. Die Überprüfungen/ Abläufe müssen anhand von Ausgaben nachvollziehbar sein. Ermöglichen Sie die Produktion eines Fehlerfalls (Vertauschung der Reihenfolge, Paketverlust etc., negative Quittungen sollen allerdings NICHT verloren gehen).
Fertigen Sie für die unterschiedlichen Abläufe Weg-Zeit Diagramme an unter Beachtung des zeitsynchronen Systems! Nutzen Sie dazu die Zustandsdiagramme (APL-Hinweise.pdf).

Hinweis:
Wir wollen für die APL vereinfachend ein quasi **zeitsynchrones System** annehmen, d.h. zu Beginn eines Zeitschlitzes wird, falls es das Sendefenster erlaubt, *EIN* neues Paket versendet (möglicherweise auch eine wiederholte Übertragung im Falle eines NACK – diese haben immer Vorrang!). Die Pakete müssen mit dem Senden gespeichert werden z.B. in einem Array von Paketen entsprechend ihrer SeqNr = Paketnummer. Mit dem Senden eines Pakets wird zudem ein Timer gestartet (z.B. Timervalue: 3fache eines Intervalls).
Die Timer mit den dazugehörigen Sequenznummern der Packet werden zeitlich relativ zueinander in einer einfachen Liste gehalten d.h. der erste Timer in der Liste ist der mit dem geringsten Wert, der tatsächliche Wert des zweiten Timers ergibt sich aus der Summe des vorigen und seines eigenen gespeicherten Wertes usw. Mit dem Verstreichen eines Intervalls muss demzufolge nur der erste Timer dekrementiert werden (siehe auch APL_Hinweise.pdf)
Dann ruht der Sender, bis entweder das Intervall endet (z.B. 300 ms) oder *EINE* Negative Quittung eintrifft. Dies kann einfach mit der Systemfunktion *select*(...) realisiert werden. Die Rückgabewerte signalisieren, welches Ereignis eingetreten ist. Im Falle eines Empfangens muss der Wert der negativ quittierten Sequenznummer überprüft (liegt diese im aktuellen Sendfenster?!) und ggf. erneut gesendet werden (Paket Timer neu setzen!). Das Sendefenster kann erst verschoben werden, wenn die Timer gesendeter Pakete verstrichen sind (ohne weitere NACKs), allerdings nur, wenn diese in Sendereihenfolge vorliegen (andernfalls müssen Timeouts im Sendefenster vermerkt werden).
Die verbleibende Zeit (Zeitmessungen!) bis zum Intervallende kann der Sender dann ruhen (Systemfunktion *sleep*()).

Abgabe:
Gliedern Sie Ihren Quellcode gemäß in die verschiedenen Schichten:
Anwendung: File-Verarbeitung, zeilenweise Übergabe an/von SR-Protokoll;
SR-Protokoll: Verwaltung der Sequenznummern, Sendefenster und Timer, NACK, retransmisions; Verwendung Transport/Netzwerkschicht zur Nutzung der UDPv6-Multicast-Kommunikation;Transfer der SR-Nachrichten via UDP Socket-Aufrufe).
Benennen Sie die Funktionen des SAP entsprechend (z.B. ToApp , FromApp etc.).

Arbeiten Sie ohne Threads – lediglich mit einer Simulation in Form von Ereignissen wie oben beschrieben, deren entsprechender Verwaltung und Abarbeitung. Ein Client kommuniziert lediglich

mit maximal 3 Multicast-Servern. Pro Anwendungsinstanz ist jeweils nur ein Socket zu etablieren und zu warten.

Testen Sie Ihre Client-Server Anwendung mittels simulierten Fehlern, günstiger Weise mit Hilfe eines *Programmarguments.*

Die Lösungen sind in Form von gut strukturiertem und kommentiertem Quellcode auf den Git-Server der Fakultät Informatik bis spätestens 18.1. hochzuladen. **Eine kurze Erläuterung den Aufruf von Client und Server betreffend, die Weg-Zeit-Diagramme als auch die Erklärung auf die selbstständige Anfertigung unter Zuhilfenahme der angegebenen Quellen!!!! als PDF müssen mit enthalten sein**. Die Lösungen müssen unter im Labor S311 entweder unter Windows oder Ubuntu lauffähig sein. Als Entwicklungsumgebung steht Visual Studio bzw. Visual Studio Code zur Verfügung.
Die Abnahme erfolgt spätestens in der letzten Vorlesungswoche während der Praktikumszeiten am Rechner in Form einer Vorführung. Der Code muss im Labor übersetzbar sein.