Praktikum Simulative Untersuchung von Protokollfunktionen des Internet

Aufgabenblatt 8 [1]

Vorbereitung

Aufgabe 1: Fehlerkontrolle
Bei der Fehlerkontrolle unterscheidet man zwischen drei grundsätzlichen Arten von Fehlern, welche sind dies?

Aufgabe 2: Hamming-Distanz
Was versteht man unter der Hamming-Distanz? Welche Hamming-Distanz müssen gültige Nachrichten mindestens aufweisen, damit ein Empfänger fehlerhafte Nachrichten mit bis zu n Bitfehlern als solche erkennen kann? Welche Hamming-Distanz ist zur Korrektur solcher Fehler erforderlich?

Aufgabe 3: Send-and-Wait
Zu welcher Klasse von Fehlerkorrekturverfahren ist das Send-and-Wait Protokoll zuzuordnen? Wie funktioniert es? Kann Send-and-Wait auch neue Probleme verursachen?

Praktischer Teil

Denken Sie bitte –wie immer bei den praktischen Aufgaben– an aussagekräftige Debug-Ausgaben. Die Sourcen für diese Aufgabe befinden sich im Verzeichnis `protsim08`.

Ihre Aufgabe in diesem Termin ist die Implementierung des Fehlerkontrollprotokolls Send-and-Wait. Hierfür müssen Sie einen Sender und einen Empfänger implementieren. Die zu ergänzenden Dateien sind `userapps/SendAndWaitSender.cc` und `userapps/SendAndWaitReceiver.cc`.

Aufgabe 4: ARQ Network
Machen Sie sich zunächst mit Hilfe der GUI und der Datei `networks/ARQNetwork.ned` mit dem Netzwerk dieser und der nächsten Aufgaben vertraut. Das Netzwerk besteht im wesentlichen aus zwei Teilnetzen, deren Hosts sternförmig mit jeweils einem Router verbunden sind. Die beiden Router der Teilnetze sind über einen einzelnen Link verbunden. Auf jedem Host läuft zur Generierung von Hintergrundlast eine `PingPongAppl`. Auf dem ersten Host eines jeden Teilnetzes befinden sich desweiteren ein `SendAndWaitSender` und ein `SendAndWaitReceiver`, der dem entsprechenden Gegenüber des anderen Teilnetzes Daten schickt (bzw. diese empfängt).

Aufgabe 5: Send-and-Wait Sender
Der Sender soll jeweils ein Datenpaket (`ARQData`) absenden und auf die Bestätigung (`ARQAck`) dieses Pakets warten. Kommt innerhalb einer gewissen Zeit keine Bestätigung (stattdessen die Timeout Message), soll das Datenpaket noch einmal geschickt werden. Sofort nach Ankunft der Bestätigung, soll das nächste Paket gesendet werden.

Obwohl der Send-and-Wait-Mechanismus selbst keine Sequenznummern benötigt, soll das Datenpaket zum einfacheren Nachvollziehen (der besseren Identifizierbarkeit jedes Pakets) eine aufsteigende Sequenznummer enthalten. **Wichtig:** Wenn Sie ein Paket wegen fehlender Bestätigung nochmal senden, soll es natürlich das **gleiche** Paket wie beim vorherigen Senden sein. Der Ziel-Knoten und die Ziel-Applikation sollen anhand der Parameter `destNode` und `destAppl` gesetzt werden. Die ID des Pakets soll `eDATA` sein, zudem muss die Methode `setKind(eNETWORK_PACKET)` des Pakets aufgerufen werden.

[1] Stand: 12. Oktober 2023

Die Größe des Pakets und die Länge des Timeouts sollen entsprechend der Parameter `packetSize` (in Bits) und `timeoutPar`, die bereits in die gleichnamigen Variablen kopiert wurden, gesetzt werden.

Aktualisieren Sie bitte auch jeweils die Zähler `sentBytes` und `ackBytes` der total gesendeten Bytes sowie der bestätigten Bytes.

Aufgabe 6: Send-and-Wait Empfänger
Der Empfänger soll für jedes korrekt empfangene Datenpaket ein Acknowledgement vom Typ `ARQAck` mit der ID `eACK` und der Größe `ackSize` (in Bits) senden. Auch hier muss die Methode `setKind(eNETWORK_PACKET)` aufgerufen werden. Fehlerhafte Datenpakete (also Pakete mit Bitfehlern, `hasBitError()`) sollen verworfen werden. In den Zählern `totalBytes` und `goodBytes` soll die Anzahl der insgesamt empfangenen Bytes bzw. die Anzahl der Bytes in neu empfangenen (also nicht duplizierten) Paketen gezählt werden. Im Zähler `duplications` soll die Anzahl der duplizierten Pakete gezählt werden.

Aufgabe 7: Durchführung von Messungen
Für Sie vorbereitet wurden verschiedene Runs mit unterschiedlichen Hintergrundlasten und Verzögerungen der WAN (Wide Area Network) Verbindung:

Run	WAN Delay	Ping-Intervall
1	5ms	1.8ms
2	5ms	1.7ms
3	5ms	1.6ms
4	10ms	1.7ms
5	15ms	1.7ms

Führen Sie die Runs mit `./runAll.sh` aus und erfassen Sie die Anzahl der empfangenen Nutzdaten (`goodBytes`) und die Anzahl der gesamten empfangenen Bytes (`totalBytes`) im Empfänger mit `./printres.sh`. (printres gibt dabei lediglich die `sendandwait??.sca` Dateien formatiert aus)

Welche Abhängigkeiten beobachten Sie und wie sind diese zu erklären? Beachten Sie, dass für die Hintergrundlast das mittlere Ping-Intervall in den Lastgeneratoren angegeben wurde; niedrigere Zahlen entsprechen also höherer Last.