Praktikum Simulative Untersuchung von Protokollfunktionen des Internet

Aufgabenblatt 9 [1]

Vorbereitung

Aufgabe 1: Effizienz von ARQ-Verfahren
Berechnen Sie den Parameter `a` für die Effizienzberechnung von ARQ-Verfahren für eine Frame-Länge von 8000 Bits, eine Bandbreite von 10Mbps und eine Ausbreitungsverzögerung von 6 ms, 11 ms und 15 ms.

Wie groß muss die Fenstergröße bei Go-Back-N in diesen Fällen mindestens sein, um maximale Effizienz zu erreichen? Wie groß ist diese Effizienz bei einer Paketfehlerwahrscheinlichkeit von 1.6%?

Praktischer Teil

Denken Sie bitte –wie immer bei den praktischen Aufgaben– an aussagekräftige Debug-Ausgaben. Die Sourcen für diese Aufgabe befinden sich im Verzeichnis `protsim09`.

Ihre Aufgabe in diesem Termin ist die Implementierung des Fehlerkontrollprotokolls Go-Back-N. Hierfür müssen Sie einen Sender und einen Empfänger implementieren. Die zu ergänzenden Dateien sind `userapps/GoBackNSender.cc` und `userapps/GoBackNReceiver.cc`. Das Go-Back-N-Protokoll, das Sie implementieren sollen, ist eine einfachere Variante von dem, welches in der Vorlesung vorgestellt wurde.

Aufgabe 2: Go-Back-N Sender
Der Sender soll folgende Eigenschaften besitzen:

- Im Abstand von `interval` Sekunden soll jeweils ein Datenpaket (`ARQData`) mit der Sequenznummer `nextSendSeqNo` abgesendet werden. Die Funktion `getDataPacket(long seqNo)` erzeugt ggf. ein passendes Paket und legt es in einem Puffer ab oder kopiert ein schon im Puffer existierendes Paket.
- Es dürfen maximal `window` unbestätigte Pakete versendet werden.
- Wenn der Timeout für das älteste gesendete Paket abläuft, sollen alle bisher noch nicht bestätigten Pakete nochmal gesendet werden.
- Wenn ein positives Acknowledgement (also mit einer größeren Sequenznummer als `lastAckSeqNo`) ankommt, sollen die Puffer für ältere Pakete mittels `freeBuffers(long start, long end)` freigegeben, ggf. ein wegen Erreichen des Fensters gestopptes Senden wiederaufgenommen und Timer und Sequenznummern aktualisiert werden.
- Ältere Acknowledgements sollen ignoriert werden.
- Zur Verwaltung der Sequenznummern können die Variablen `lastAckSeqNo` und `nextSendSeqNo` verwendet werden. Beachten Sie bitte, dass `lastAckSeqNo` immer die Sequenznummer des vom Empfänger als nächstes erwarteten Pakets enthalten soll.
- Zum Schedulen von Timeouts und Senden von Paketen sind die Variablen `nextSendTime` und `timerExpiration` gedacht. Die benötigten Timer sind in `nextSendTimer` und `expirationTimer`.
- Timeouts für die einzelnen Pakete können in `packetBuffer[seqNo].timeout` verwaltet werden.

[1] Stand: 12. Oktober 2023

- Aktualisieren Sie bitte jeweils die Zähler `sentBytes` und `ackBytes` der total gesendeten Bytes sowie der bestätigten Bytes (beachten Sie bitte, dass eine Acknowledgement bei Go-Back-N mehrere Pakete bestätigen kann).
- Zeichnen Sie die Sequenznummern der gesendeten Pakete und der empfangenen Acknowledgements in den `cOutVector`en `sendSeqNumVector` und `ackSeqNumVector` auf.

Aufgabe 3: Go-Back-N Empfänger
Der Empfänger soll folgende Eigenschaften besitzen:

- Wird das als nächstes erwartete Paket ohne Fehler empfangen, soll mit `sendAck()` ein Acknowledgement gesendet werden. `sendAck()` erwartet als erstes Argument das Paket, welches bestätigt wird, als zweites die Sequenznummer des als nächstes erwarteten Pakets.
- Fehlerhafte Pakete oder Pakete außerhalb der Reihe werden ohne Reaktion gelöscht.
- Zur Verwaltung der Sequenznummern kann die Variable `lastReceivedSeqNo` verwendet werden.
- Aktualisieren Sie bitte jeweils die Zähler `totalBytes` und `goodBytes` der insgesamt empfangenen Bytes sowie der Bytes aus reihenfolgerichtig empfangenen Paketen.
- Zeichnen Sie die Sequenznummern der beschädigten oder nicht erwarteten Pakete in den `cOutVector`en `corruptedSeqNumVector` und `outOfOrderSeqNumVector` auf.

Aufgabe 4: Testen und 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	Bemerkungen
1-4	5ms	1.8ms	
5-8	5ms	1.6ms	
9-12	5ms	1.5ms	
13-16	10ms	1.8ms	
17-20	15ms	1.8ms	
21-24	5ms	1.8ms	NACK
25-28	5ms	1.5ms	NACK

Führen Sie zunächst Run1 aus und teilen sie anschließend den Outputvektor `results/gobackn01.vec` mit dem Script `splitvec.py`. Es befindet sich im `eval` Ordner. Informationen über die Benutzung erhalten Sie mit einem `./splitvec.py -h` Aufruf. Das Skript erwartet eine Auswahl der Subvektoren als Eingabe.

Nachdem Sie nun den SendSeqNo- und AckSeqNo- Subvector (z.B. Subvector 0 und 22) extrahiert haben, können Sie im `eval` Ordner das Script `./plot.sh out0 out22` mit den beiden Subvecktoren aufrufen. Es wird Gnuplot gestartet und ein Diagramm erzeugt. Es zeigt die Sequenznummern in Abhängigkeit zur Simulationszeit. Versuchen Sie die Grafik zu deuten.

Um die statistische Zuverlässigkeit der Ergebnisse zu erhöhen werden von den Experimenten jeweils 4 Runs ausgeführt. Führen Sie die Runs 1-20 aus. Zur Auswertung steht Ihnen das Skript `ps_eval_arq.py` zur Verfügung, welches als Argumente die zusammengehörenden Runs und den Basisnamen der Skalar-Dateien erwartet. Dieses Skript berechnet den Mittelwert und das 95%-Konfidenzintervall von `throughput` und `goodput` für die verschiedenen Empfänger. Da wir deutlich weniger als 30 Messungen durchführen benutzt das Skript zu diesem Zweck nicht die Normalverteilung, sondern die Student-Verteilung (auch t-Verteilung genannt).

Für den ersten Satz von Messungen muss man das Skript also wie folgt aufrufen:

```
./ps_eval_arq.py -r 1,4 ../results/gobackn
```

Welche Abhängigkeiten beobachten Sie und wie sind diese zu erklären? Bei der Hintergrundlast ist wieder das mittlere Ping-Intervall in den Lastgeneratoren angegeben; niedrigere Zahlen entsprechen also höherer Last.

Um für alle Runs von 1-20 (mit `./runAll.sh` gestartet) die Auswertung vorzunehmen existiert das Skript `eval/evalAll.py`.

Aufgabe 5: (**optional**)Negative Acknowledgements
Erweitern Sie Sender und Empfänger um negative Acknowledgements. Der Empfänger soll genau dann ein Acknowledgement mit der Sequenznummer `x` wiederholen, wenn er Paket `x+1` empfängt, obwohl er Paket `x` erwartet. Alle folgenden Pakete soll der Empfänger wie bisher ohne weitere Reaktion löschen. Negative Acknowledgements sollen nur gesendet werden, wenn `sendNack true` ist.

Der Sender soll Acknowledgements mit derselben Sequenznummer wie ein vorheriges (`lastAckSeqNo`) als negatives Acknowledgement interpretieren und im Vektor `nackSeqNumVector` aufzeichnen. Wie beim Timeout soll er das Senden seit dem zuletzt bestätigten Paket wiederholen.

Die Runs 21-28 (siehe oben) benutzen negative Acknowledgements. Lassen Sie diese bitte durch laufen und werten Sie sie aus.