Praktikum Simulative Untersuchung von Protokollfunktionen des Internet

Aufgabenblatt 6 [1]

Vorbereitung

Aufgabe 1: Queue
Notieren Sie bitte im Pseudo-Code den Algorithmus, der in der `handleMessage()`-Methode einer Queue ablaufen muss, wie sie im praktischen Teil beschrieben wird. Eventuelle Hinweise sind bereits im Quelltext enthalten.

Aufgabe 2: Statistiken in OMNeT++
Machen Sie sich mit dem OMNeT++-Klassen `cStdDev`, `cHistogram` und `cOutVector` vertraut. Diese Klassen dienen zum Sammeln statistischer Daten in OMNeT++.

Aufgabe 3: Link-Utilization
Wie kann man mittels einer zeitgewichteten Statistik effizient die durchschnittliche Link-Utilization bestimmen? Bei welchen Ereignissen müssen dazu Daten erfasst werden?

Aufgabe 4: Link-Utilization
Ein Link habe eine Bandbreite von 2 Mbps. Vier Applikationen versenden in regelmäßigen Abständen Nachrichten von 8000 bits Größe über diesen Link. Das Zeitintervall zwischen den Nachrichten ist bei allen vier Applikationen gleich. Schätzen Sie ab, wie lang das Zeitintervall mindestens gewählt werden muss damit keine Überlast auf diesem Link entsteht.

Aufgabe 5: Determinismus und Zufall
Begründen Sie die Notwendigkeit von Zufallseinflüssen bei der Simulation der Link-Ausgangswarteschlange. Auf welche Weise wird dieser Einfluss im Omnet++ realisiert? Erwarten Sie aufgrund dieser Einflüsse verschiedene Ergebnisse bei wiederholter Ausführung Ihrer Simulation mit gleichen Parametern und gleichem Code?

Praktischer Teil

Denken Sie bitte – wie immer bei den praktischen Aufgaben – an aussagekräftige Debug-Ausgaben. Die Sourcen für diese Aufgabe befinden sich im Verzeichnis `protsim06`.

Bisher konnten Sie unbegrenzt viele Pakete auf ein Link senden. Durch die begrenzte Bandbreite des Links konnten sich die Pakete unbegrenzt aufstauen. In der Realität hat ein Knoten natürlich nur begrenzten Speicher. Die Pakete für einen ausgehenden Link werden in einer Warteschlange (Queue) begrenzter Länge gespeichert. Ist diese voll, werden Pakete gedroppt. Eine wichtige Anwendung von Netzwerk-Simulation ist die Frage, wieviel Kapazität an den Knoten nötig ist, wie hoch die Verlustrate bei gegebener Queue-Kapazität ist, wie lange die Wartezeit usw.

Aufgabe 6: Queue-Implementierung
Ihre erste Aufgabe in diesem Termin ist es, eine Interface-Queue zu implementieren. Ergänzen Sie hierfür die Datei `protsim06/support/InterfaceQueue.cc`. Nachdem Sie nun schon einige Routine haben, haben wir Ihnen nur einen Rahmen vorgegeben, den Sie selbständig ausfüllen können. Ergänzen Sie Attribute, die Sie brauchen und implementieren Sie bitte `initialize()` und `handleMessage()`. Die Queue soll folgenden Anforderungen genügen:

[1] Stand: 12. Oktober 2023

- Die Queue soll als FIFO-Queue (First-in First-out) arbeiten, d.h. die Pakete sollen in der Reihenfolge, in der sie gepuffert werden auf den Link gesendet werden. Möglicherweise nützlich für Sie ist die `cQueue`-Klasse von OMNeT++.
- Die maximale Queue-Länge wird im Parameter `maxQueueLength` gespeichert.
- Wenn ein Paket ankommt und der Link frei ist, soll es sofort gesendet werden, ansonsten ist es in die Queue einzufügen.
- Wenn schon `maxQueueLength` Pakete in der Queue sind, sollen neu ankommende Pakete mit einer Debug-Ausgabe gedroppt werden.
- Wenn der Link frei wird und (mindestens) ein Paket in der Queue ist, soll dieses gesendet werden. Sie benötigen wahrscheinlich einen Timer, um dies zu erreichen.

Zur Auswertung der Simulation müssen statistische Daten gesammelt werden:

- Bei jeder Änderung der Queue-Länge soll
  - die alte und neue Queue-Länge in den `cOutVector` `queueLengthVector` eingefügt werden,
  - die neue Queue-Länge in das `cHistogram` `queueLengthHist` eingefügt werden,
  - die alte Queue-Länge gewichtet mit der Zeit seit der letzten Änderung in die `cStdDev` `queueLengthStat` eingefügt werden.
- Beim Senden einer Nachricht soll die Wartezeit der Nachricht
  - in den `cOutVector` `waitingTimeVector` eingefügt werden,
  - in die `cStdDev` `waitingTimeStat` eingefügt werden.

  Nützlich dürfte hier die Methode `getArrivalTime()` von `cMessage` sein.
- Wenn der Link seinen Zustand ändert, soll die `cStdDev` `linkUtilizationStat` aktualisiert werden.
- Zählen Sie bitte auch die Anzahl der gedroppten Pakete und geben Sie diese aus.

Aufgabe 7: Messungen
Mit der implementierten Queue können Sie nun Messungen durchführen. Es hat sich herausgestellt, dass das Interface 1 im Knoten 2 einen Flaschenhals darstellt. Daher lohnt es sich, diesen näher zu untersuchen. Beobachten Sie die Vektoren, Statistiken und das Histogramm der entsprechenden `InterfaceQueue`.

Bei den ersten drei Runs der Simulation generieren die Applikationen Nachrichten mit Ereignis-Abständen, die zufällig aus einer Exponentialverteilung (Verteilungsfunktion $y = 1 - e^{-\frac{x}{\mu}}$) gezogen werden. Der Erwartungswert variiert bei jedem Run:

Run	Erwartungswert μ
1	18 ms
2	16 ms
3	15 ms

Führen Sie jeden Run bis zu seinem Ende (30 s) durch. Beobachten Sie insbesondere die Veränderungen des Histogramms. Welche Aussagen lassen sich hinsichtlich der Lastsituation machen? Wann herrscht Überlast?

Aufgabe 8: (optional) Messungen
Bei den Runs 4 und 5 ist der Mittelwert der Zwischenankunftszeiten wie bei Run 2 16 ms. Allerdings ist er bei Run 4 konstant 16 ms; bei Run 5 folgt er einer Gleichverteilung zwischen 0 und 32 ms. Wiederholen Sie den letzten Versuch mit diesen beiden Runs. Was lässt sich hier über die Last aussagen? Können Sie sich die Unterschiede erklären? Experimentieren Sie ruhig mit unterschiedlichen Intervallen und Verteilungen. Sie können diese in den entsprechenden Run-Abschnitten von `omnetpp.ini` ändern.