Übung 9 zur Vorlesung "Vernetzte Systeme" WS 1999/2000

Prof. Dr. F. Mattern

Ausgabedatum: 15. Dez. 1999
Abgabedatum: 22. Dez. 1999

35. Aufgabe (Token-Release-Strategien)

In einem Token-Ring-Netz mit *delayed token release* wartet die sendende Station, bis die von ihr gesendete Nachricht (d.h. das erste Bit dieser Nachricht) einmal den Ring komplett umrundet hat (und wieder an der Sendestation angekommen ist), bevor sie das Token an die nächste Station weitergibt. Dem gegenüber steht das *immediate token release*, bei dem das Token sofort im Anschluss an das gesendete Paket auf den Ring gelegt wird.

a) (2 Punkte) In FDDI Netzen wird das *immediate token release* Verfahren verwendet. Warum?

b) (2 Punkte) Unter welchen Umständen macht es keinen Unterschied, welches der beiden Verfahren verwendet wird?

36. Aufgabe (Ringpuffer in Java)

Nachdem Sie sich in Aufgabe 34 Gedanken über die Anforderungen einer Ringpuffer-Implementation des Empfangsfensters im Sliding-Window-Protokoll gemacht haben, sollen Sie dies nun in einem einfachen Java-Programm umsetzen.

a) (6 Punkte) Laden Sie sich dazu von der Vorlesungs-Web-Seite die Grundstruktur einer Ringpuffer-Klasse (`Ringpuffer.java`) herunter und implementieren Sie die darin vorgegebenen Methoden entsprechend den in Aufgabe 34 gegebenen Anforderungen:

`http://www.inf.ethz.ch/department/IS/vs/education/WS9900/VS/`

Folgende Methoden sollen implementiert werden:

- `Ringpuffer (int size)`
  Der Konstruktor der Klasse. Erzeugt einen Ringpuffer der Grösse `size`.
- `int idx (int s)`
  Liefert die Index-Position im Puffer-Array für eine gegebene Sequenznummer `s`.

- `Object getnextpacket()`
  Liefert das nächste Datenpaket in der Sequenz zurück. Falls das nächste Paket noch nicht im Puffer gespeichert wurde, soll `null` zurückgeliefert werden.

- `int storepacket (int s, Object data)`
  Speichert gegebenenfalls den Inhalt `data` des Datenpaketes mit der Sequenznummer `s` im Array und liefert die soeben erhaltene Sequenznummer als ACK zurück. Falls kein ACK gesendet werden soll, muss `-1` zurückgegeben werden.

Testen Sie dann Ihre Ringpuffer-Implementation mittels eines kleinen Testprogramms (`buffertest.java`), welches ebenfalls von der Web-Seite der Vorlesung heruntergeladen werden kann. Das Testprogramm schickt eine Beispielsequenz von Wörtern an das durch den Ringpuffer simulierte Empfangsfenster und liest es in unregelmässigen Abständen aus (es simuliert also praktisch die darüber und darunter liegenden Protokollschichten).

Schicken Sie die erweiterte Version der Ringpuffer-Klasse Ihrem Tutor als E-Mail-Attachment. Bitte beachten Sie, dass keinerlei Veränderungen an dem Testprogramm nötig sein sollten, um Ihre Ringpuffer-Implementation zu testen!

b) (4 Bonuspunkte) Für alle diejenigen, die noch Lust haben und ein paar Extra-Punkte sammeln wollen, hier noch eine optionale[1] Erweiterung der Ringpuffer-Klasse:

Implementieren Sie zusätzlich die Methode `show()`, die eine String-Repräsentation des momentanen Zustandes des Ringpuffers zurückliefert. Folgende Informationen sollten aus der (kompakten, einzeiligen) Repräsentation leicht ablesbar sein:

- Momentane Position von *Nfe*.
- Belegt-Anzeige für einzelne Pufferplätze (ob leer oder zum Auslesen bereit).
- Momentane Sequenznummern der einzelnen Pufferplätze.

Ein Beispiel für solch eine Repräsentation wäre:

```
8   9  *[5]   6   [7]
```

Hier ist die Position des *Nfe* mit einem * gekennzeichnet, die Zahlen geben die jeweilige Sequenznummer des Pufferplatzes an, zum Auslesen bereite (d.h. belegte) Speicherplätze sind mit `[]` Klammern umgeben. Im Beispiel wären also an der 3. und 5. Position Pakete im Puffer mit den Sequenznummern 5 bzw. 7. Das nächste ausgelesene Paket wäre 5 (an der 3. Position), während die Pakete mit den Sequenznummern 6, 8 und 9 noch ausstehen.

$Id: uebung9.lyx,v 1.10 1999/12/15 14:25:22 langhein Exp $

[1] Die 4 Punkte zählen also nicht in der "Gesamtwertung" zum Testat - wer diese Aufgabe nicht bearbeitet, verliert also keine Prozentpunkte - sie können aber zur Aufbesserung des Punktekontos verwendet werden.