9. Übung zur Vorlesung "Vernetzte Systeme" WS 2000/2001

Prof. Dr. F. Mattern

Ausgabedatum: 20. Dez. 2000
Abgabedatum: 10. Jan. 2001

Aufgabe 35 (Bluetooth)

Bei *Bluetooth* handelt es sich um ein System zur spontanen, drahtlosen Kommunikation über kurze Entfernungen ($\approx$ 10m). Bluetooth wird sich voraussichtlich in den nächsten Jahren stark bei portablen Geräten verbreiten.

Die kleinste Einheit eines Bluetooth-Systems ist ein sogenanntes *Piconetz*. Dieses besteht aus einem Master und bis zu sieben Slaves.[1] Der Master gibt den Takt vor und entscheidet, welche Station senden darf. Innerhalb eines Piconetzes können daher keine Kollisionen auftreten.

Probleme können auftauchen, wenn sich mehrere Piconetze zu einem sogenannten *Scatternet* überlappen. Die Piconetze untereinander sind nicht synchronisiert, es handelt sich hierbei um ein dezentrales System, in dem Kollisionen auftreten können.

Um diesem Problem entgegenzuwirken, wird bei Bluetooth das sogenannte *Frequenzsprungverfahren ("Frequency Hopping")* eingesetzt: es gibt 79 verfügbare Frequenzkanäle, auf denen gesendet werden kann.[2] 1600 mal in der Sekunde wird der Frequenzkanal nach einem pseudozufälligen Muster neu bestimmt. Da jedes Piconetz seine eigene (allen Stationen im Piconetz bekannte) Frequenzreihenfolge hat, ist somit die Wahrscheinlichkeit, dass sich zwei Piconetze beim Senden stören, gering. Wenn eine Kollision durch Benutzung gleicher Frequenzen dennoch auftritt, so wird dies bemerkt und die entsprechenden Pakete werden erneut gesendet.

Für die Aufgabe nehmen Sie bitte an, dass in allen Piconetzen in jedem Zeitschlitz gesendet wird und dass pro Zeitschlitz genau ein Paket gesendet wird, welches den Zeitschlitz ausfüllt.

a) (2 Punkte) Wie gross ist die Wahrscheinlichkeit, dass für ein gesendetes Paket eine Kollision auftritt, wenn sich zwei Piconetze überlappen?[3]

b) (2 Punkte) Wie gross ist diese Wahrscheinlichkeit bei 10 sich überlappenden Piconetzen?

c) (3 Punkte) Ein Paket wird direkt im anschliessenden Zeitschlitz vom Empfänger bestätigt. Auch solche Acknowledgement-Pakete können durch Störung eines anderen Piconetzes verloren gehen. Berechnen Sie Teilaufgabe a) und b) neu unter der Annahme, dass ein Paket nur dann als erfolgreich gesendet gilt, wenn sowohl das Paket selbst als auch das Ack-Paket nicht gestört werden.

d) (3 Punkte) Berechnen Sie Teilaufgabe a) und b) neu unter folgender Annahme: alle Piconetze des Scatternetzes sind bezüglich des Zeitpunktes, wo ein Frequenzwechsel stattfindet, synchronisiert.[4]

[1] Da Bluetooth ein *spontanes* System ist, kann jede Station potentiell Master sein. Dieser wird spontan und nur für kurze Zeit bestimmt.

[2] Diese sind jeweils 1MHz breit und erstrecken zwischen 2.402 und 2.480 GHz

[3] Beachten Sie bitte dabei, dass die Zeitschlitze der einzelnen Piconetze zwar gleich lang, aber nicht *synchron* sind.

[4] Ignorieren Sie dabei die Acknowledgement-Pakete

Aufgabe 36 (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/vs/education/WS0001/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[5] 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.

[5] 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.