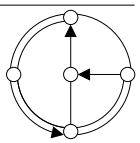
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2007 Prof. R. Wattenhofer / Michael Kuhn / Remo Meier

Vernetzte Systeme
Lösungsvorschlag 8

1 Fragmentierung von IPv4 Paketen

a) Für das Length-Feld stehen 16 Bits zur Verfügung, für das Offset-Feld nur 13. Daher lässt sich mit dem Offset nicht jedes einzelne Byte der Nutzdaten adressieren, sondern nur jeder Block der Grösse $2^3 = 8$ Bytes.

Im Extremfall könnte ein Netz somit eine MTU von 28 Bytes haben. Sollte über ein solches (hypothetisches) Netz ein maximal grosses IP-Paket (65535 Bytes) übertragen werden, so würde es in $2^{16}/2^3 = 2^{13} = 8192$ Fragmente aufgeteilt werden. Jedes Fragment würde 8 Byte Nutzdaten, das letzte nur 7 Byte Nutzdaten tragen.

b) Die Nachricht ist 1000 Bytes lang. Durch den UDP-Header kommen 8 Bytes hinzu, so dass 1008 Bytes als IP-Nutzdaten übertragen werden müssen. Der Sender fragmentiert das Ausgangspaket, um die MTU auf Netz 1 (1024 Bytes) nicht zu überschreiten. Folgende Pakete kommen beim Router an:

Length	Offset	MF
1000 + 20 = 1020	0	1
8 + 20 = 28	125	0

Der Router fragmentiert das 1020-Byte-Paket erneut, um die MTU von Netz 2 (512 Bytes) nicht zu überschreiten. Beim Empfänger kommen schliesslich diese Pakete an:

Length	Offset	MF
488 + 20 = 508	0	1
488 + 20 = 508	61	1
24 + 20 = 44	122	1
8 + 20 = 28	125	0

c) Angenommen, die Netzwerkschicht auf Senderseite würde das UDP-Datagram auf mehrere IP-Pakete aufteilen. Dann würde beim Empfänger jedes IP-Paket für ein eigenständiges UDP-Datagram gehalten werden (Protocol = 17 im IP-Header). Beide würden jedoch wahrscheinlich verworfen werden, da die Längenangabe im UDP-Header und die Prüfsumme falsch ist. Eines der Pakete würde möglicherweise gar nicht dem richtigen Empfängerprozess zugeordnet werden, da der "Zielport" irgendwelche Werte enthalten würde, die gerade an dieser Stelle in den Nutzdaten standen.

2 Cyclic Redundancy Check

a) Um die Checksumme zu berechnen, muss zuerst das vollständige T als $D + EDC$ bestimmt werden. Die Länge des $EDCs$ entspricht dem Grad von $G(x)$. Somit muss 1101011110010000 : 10011 berechnet werden:

```
1101011110010000 : 10011 = 110000111101 Rest 0111
10011
-----
 10011
 10011
 -----
  000011100
      10011
      -----
       11111
       10011
       -----
        11000
        10011
        -----
         10110
         10011
         -----
          010100
           10011
           -----
            0111 = Rest
```

Übertragen wird also $T = 1101011110010111$.

b) Da, wie die folgende Rechnung zeigt, T' ohne Rest durch G teilbar ist, kann der Fehler nicht erkannt werden.

```
1100010010010111 : 10011 = 110100111101
10011
-----
 10111
 10011
 -----
  010000
   10011
   -----
    0011100
      10011
      -----
       11111
       10011
       -----
        11000
        10011
        -----
         10111
         10011
         -----
          010011
           10011
           -----
            0000 -> Kein Rest!
```

c) Wie im Skript auf Folie 5/18 steht, muss ein Fehler $E(x)$ so beschaffen sein, dass er ohne Rest durch $G(x)$ teilbar ist, um **nicht** erkannt werden zu können. Bezogen auf den *speziellen* Modulo-/Divisionsoperator bedeutet das, dass man G beliebig oft an beliebigen Stellen mit dem Ursprungsstring T *xor*en kann, ohne das sich das Ergebnis der Division ändern würde. Zum Beispiel würde keiner der folgenden Strings als Fehler erkannt werden:

```
  1101011110010111 -> T aus a)
+    10011
  ----------------
  1100010010010111 -> T' aus b)
+            10011
  ----------------
  1100010010000100
+ 10011
  ----------------
  0101110010000100 ...
```

3 Slotted Aloha

a) Die Wahrscheinlichkeit (WK), dass der Server erfolgreich sendet, ist gegeben durch die WK, dass er tatsächlich sendet, multipliziert mit der WK, dass kein Client versucht zu senden:

$P_s = Pr[Server\ succeeds] = p_s * (1-p)^n$

Die WK, dass ein bestimmter Client sendet, ist gegeben durch die WK, dass dieser Client tatsächlich sendet, multipliziert mit der WK, dass weder die anderen Clients, noch der Server versuchen zu senden:

$P_1 = Pr[One\ of\ n\ Clients\ succeeds] = p * (1-p)^{(n-1)} * (1-p_s)$

Schliesslich ist die WK, dass irgendeine Station erfolgreich sendet, gegeben durch:

$P = Pr[Any\ Station\ succeeds] = n * P_1 + P_s$

Da wir die WK p für die Clients optimieren wollen, müssen wir analog zu Folie 5/39 P ableiten, Nullsetzen und anschliessend nach p auflösen. Unter Maple sieht das wie folgt aus ($p0$ ist die geforderte Antwort der Aufgabe):

```
> P := n*p*(1-p)^(n-1)*(1-ps)+1*ps*(1-p)^n;
```

$$P := n\,p\,(1-p)^{(n-1)}\,(1-ps) + ps\,(1-p)^n$$

```
> P_ := diff(P,p);
```

$$P\_ := n\,(1-p)^{(n-1)}\,(1-ps) - \frac{n\,p\,(1-p)^{(n-1)}\,(n-1)\,(1-ps)}{1-p} - \frac{ps\,(1-p)^n\,n}{1-p}$$

```
> p0 := solve(P_ = 0,p);
```

$$p0 := \frac{-1+2\,ps}{-n+n\,ps+ps}$$

b) Sie erhalten den Wert von etwa 5.3%, in dem Sie die vorgegebenen Werte in die unter a) berechnete Formel einsetzen:

```
> evalf(subs(ps=1/3,n=10,%));
```

$$0.05263157895$$

Beachten Sie, dass die Sende-Wahrscheinlichkeit p der Clients für $p_s \geq 0.5$ Null oder sogar negativ werden kann! Das heisst, dass es in diesem Fall besser wäre, wenn die Clients gar nicht senden.

c) Setzen Sie zuerst die Lösung für $p0$ in die ursprüngliche Formel für P ein:

```
> simplify(subs(p=p0, P));
```

$$-\frac{(-n+n\,ps+ps)\,(\frac{-n+n\,ps+1-ps}{-n+n\,ps+ps})^n}{n-1}$$

Berechnen Sie anschliessend den Wert von etwa 41% durch Einsetzen der in b) gegebenen Werte:

```
> evalf(subs(ps=1/3, n=10, %));
```

$$0.4098065338$$

d) Das *normale* Slotted Aloha liefert für 11 gleichberechtigte Stationen die optimale Sende-Wahrscheinlichkeit $p_n = 1/11$ für jede Station. Für die Wahrscheinlichkeit, dass irgendeine Station senden kann, ergibt sich mit der Formel $P_n = (1-\frac{1}{n})^{n-1}$ aus dem Skript der Wert $P_n \approx 0.386$.

Die Effizienz der Kanalnutzung steigt somit leicht an, allerdings dürfen die Clients nur noch mit einer geringeren Wahrscheinlichkeit senden.

e) Bei einem Reservations-Verfahren kann die Effizienz der Kanalbelegung 100% betragen. Das ist z.B. der Fall, wenn bei n Stationen der Kanal in n Slots eingeteilt wird, wobei jeder Slot von einer anderen Station verwendet wird und diese jedesmal sendet (also genügend Daten vorhanden sind, ansonsten bleibt der Slot ungenutzt).

Dabei ist zu beachten, dass die Einteilung statisch ist: Wird eine Station aus dem System genommen, so bleiben ihre Slots ungenutzt; kommt eine Station hinzu, kann diese nicht senden, weil kein Slot frei ist, oder sie nicht weiss, welcher. Hingegen kann beim Slotted Aloha-Verfahren die tatsächliche Anzahl der Stationen durchaus gravierend von der für das System optimalen (anhand derer die Sende-Wahrscheinlichkeit bestimmt wurde) abweichen: Die Effizienz sinkt dann zwar, aber immerhin können alle Stationen senden (siehe Folie 5/41). Mit adaptiven Verfahren kann diese sogar während des Betriebs dezentral maximiert werden.

In einer statischen Umgebung sollte man daher ein Reservations-Verfahren einsetzen, da deren Effizienz deutlich höher liegen kann. Hingegen sollte in einem dynamischen System das Slotted Aloha-Verfahren eingesetzt werden. Interessant sind auch hybride Systeme, in denen etwa ein Slotted Aloha-Kanal für die Reservierung von Slots auf einem anderen Kanal eingesetzt wird. Ein solches Verfahren kommt z.B. bei GSM zum Einsatz.

4 Verschachtelung von Paketen

a) Die ersten 22 Bytes gehören zum Ethernet-Frame, danach folgen jeweils 20 Bytes für den IP- und den TCP-Header. Das Ethernet-Frame kann man an der typischen Form der Präambel (1010...) erkennen. Der Wert des *Type*-Felds (0x0800) lässt darauf schliessen, dass ein IP-Paket eingebettet ist. Und das TCP-Paket erkennt man daran, dass das Protokoll-Feld des IP-Headers den Wert 6 hat.

Ethernet:

bytes

7	1	6	6	2
Preamble 0x AA AA AA AA AA AA AA	SFD 0x AB	DA 00:90:27:A2:C3:A7	SA 00:D0:BC:EC:AA:64	Type 0x 08 00

IP:

bits

0	4	8	16	19	31
Version 4	HL 5	Type of Service 0	Total Length 0x 4B (75)		
Identification 0x 87 06 (34566)			Flags 010	Fragment Offset 0	
Time to Live 0x FE (254)		Protocol 6	Header Checksum 0x 62 C3 (25283)		
Source Address 129.132.13.66					
Destination Address 129.132.130.152					

TCP:

bits

0 4 10 16 31

Source Port 0x 00 6E (110)								Destination Port 0x 0A DA (2778)
Sequence Number 0x 78 20 5C 99 (2.015.386.777)								
Acknowledgement Number 0x 93 F6 04 FB (2.482.373.883)								
DO 5	Reserved 0	U 0	A 1	P 1	R 0	S 0	F 0	Window Size 0x 22 38 (8760)
Checksum 0x CE CA (52938)								Urgent Pointer 0

b) Die eigentlichen Daten schliessen sich an das TCP-Paket an. Die Länge beträgt 35 Bytes. Diesen Wert kann man z.B. bestimmen, indem man von der *Total Length* (75) die Länge des IP-Headers (*Header Length/HL* (5*4=20)) und die des TCP-Headers (*Data Offset/DO* (5*4=20)) abzieht.

Im Klartext lässt sich: '+OK Pop server at vs signing off.' (gefolgt von *Carriage Return* und *Line Feed*) erkennen.

c) Abgesehen von dem in b) beschriebenen 'Pop server', kann man auch aus dem angegebenen Sourceport 110 schliessen, dass es sich wahrscheinlich um ein Mailprogramm handelt.

d) Die IP-Checksumme wird über den IP-Header berechnet, wobei das Checksummenfeld selbst nicht beachtet wird:

```
  4500
+ 004B
  ----
  454B
+ 8706
  ----
  CC51
+ 4000
  ----
 10C51
 -----  -> \"Ubertrag aufsummieren
  0C52
+ FE06
  ----
 10A58
 -----  -> \"Ubertrag aufsummieren
  0A59
+ 8184
  ----
  8BDD
+ 0D42
  ----
  991F
+ 8184
  ----
 11AA3
 -----  -> \"Ubertrag aufsummieren
  1AA4
+ 8298
  ----
  9D3C  -> 10011101 00111100 -> Komplement/Negation -> 01100010 11000011 -> 62C3
```