ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Johannes Bader

Selfishness in Wireless Medium Access

diploma thesis

Semester:	10
Betreuer:	Thomas Moscibroda
	Pascal von Rickenbach
Abgabedatum:	10.10.2006

TIK Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Contents

Chapter 1

The ALOHA protocol

1.1 Description

1.1.1 History

ALOHA - which means "hello" in the Hawaiian language - is a protocol from the area of the computer networks and describes a media access control method (therefore being part of second layer of the OSI model [Tan03]), which means the protocol allows multiple participants to use the same shared medium for transmission. The protocol was developed by Norman Abramson at the university of Hawaii in 1970 for the so called ALOHAnet (see section 1.1.3). It was originally meant for wireless networks but his core concept is the base for wire communication systems such as the very popular Ethernet.

1.1.2 Protocol description

The ALOHA–protocol is carried out as follows [Tan03], [vM06]:

Algorithm 1 ALOHA-protocol

The users can send whenever they have new data to dispatch.
If collisions occur the user waits for a random amount of time and resends the packet thereafter. He repeats this step as long as the packet collides.

That a sent package collided, the station either finds out by interception of the channel or, if this is not possible, by a feedback from a central hub (in the ALOHAnet, the nodes listened to see if the message they sent was sent back to them by the hub located at the university. If they never got the packet back this meant it collided with an other packet). All packets are of the same length since the throughput is maximized this way. As soon as more than one participant uses the channel, a collision occurs and all sent packets have to be dropped. It is sufficient on that occasion if the last bit of one of the packets overlaps with the first bit of an other packet - the checksum can only distinguish between collision and successful transmission.

Each user can be in three states, shown in figure 1.1 :

Idle: The user doesn't have any data he wants to send. As soon as a packet is generated and available, he sends it.

Send: The user has sent a packet and waits for the feedback. If the packet was transmitted successfully, he goes to the state idle thereafter. If, on the other hand, a collision occurred he changes to the blocked state.

Blocked: A collision occurred so the user holds sending for a random amount of time before retrying to transmit the collided packet.

1.1.3 ALOHAnet

The so called ALOHAnet connected many islands around Hawaii with the university of Honolulu.

Figure 1.2 shows the setting: Five Islands A to E and the University U want to communicate with

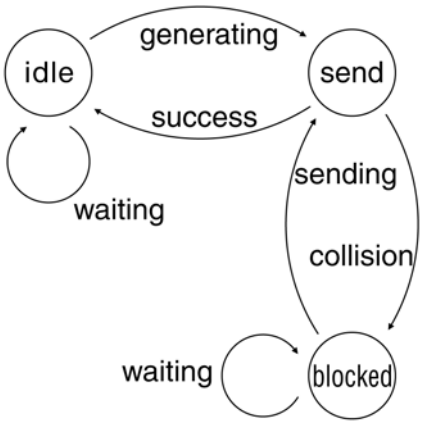


Figure 1.1. This state diagram shows the course of the ALOHA-protocol. As long as no packet is available, the user waits in the *idle* state. As soon as a packet is generated, the player tries sending it. Upon success, he goes back to the idle state. When the packet has collided on the other hand, he changes to the blocked state and waits a random amount of time until trying to the send packet again.

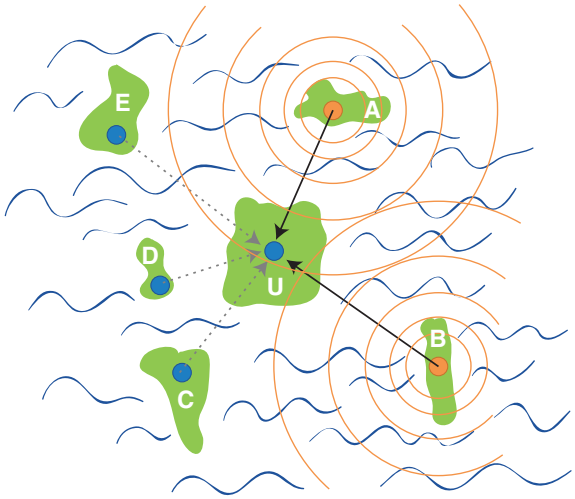


Figure 1.2. Shows the star-shaped arrangement of the participants in the ALOHAnet. The island A to E want to communicate with each other, they send their data first to the central hub located at the university U, which thereafter redirects the packet to the final destination. The central hub acknowledges every packet it receives using a dedicated channel. As soon as more than one node sends (A and B in the figure), a collision occurs and both packets are lost.

each other. All Islands, the nodes of the system, can send data any time they want using packet radio, but as soon as at least two of them send at the same time, the signals are ruined, since all nodes use the same frequency.

Two avoid the problem, a user occupies the radio for a very long time, the data had to be broken down to small packets. The network is star-shaped, which means, all users have to send their data to the central hub located at the university, which redirects the data to the final destination. The networks used two frequency channels:

Broadcast channel This channel was used to transmit the packets from the central hub (U in the example) back to the nodes (A-E). Using a dedicated channel for these messages guarantees, that no feedback is ruined or lost due to collisions.

Random access channel This channel was used to transmit the data from the nodes (A-E) to the central hub at the university (U).

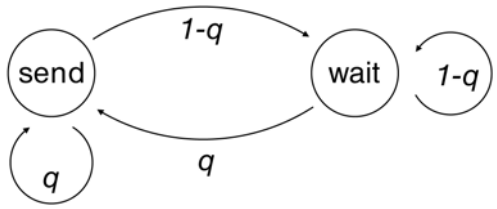


Figure 1.3. Course of the simplified ALOHA protocol when assuming binominal transmission attempts and infinitly large buffers. Each participant has data he wants to transmit anytime and sends in a time interval T with probability q, either a new packet or one that collided in a previous attempt.

The nodes used modems with a baud rate of 9600 symbols per second and packets 146 bits long, containing 80 bits payload.

1.2 Mathematical models

In this section the mathematical models to calculate the characteristics of the ALOHA protocol are given.

1.2.1 Distribution of the transmission attempts

For the following calculations, two simplified models are considered:

Model 1 (binominal): There are N participants taking part in the protocol. Every one of them has *always* a packet that he wants to send anytime. That means, he generates at least as many packets as he transmits and the buffer size is sufficiently large. (We will explain the buffer model in greater detail in section 1.2.3). Let the probability, that a specific users sends within the time interval T when he has a packet pending be q. This model is used for example in [vM06]. .

Model 2 (poisson): The number of participants is very large ($N \gg 1$) and the number of transmission-attempts during the time T is a Poisson process with mean value G, called the *offered load*. This model is very common (for example used by [Tan03]).

1.2.2 Processing packet when in the idle state

If a user is at the states send or blocked , further incoming packets are either buffered of discarded, depending on the buffer space available (see section 1.2.3 for an in depth analysis of the impact of buffer size). We make use of the following models throughout the paper:

No buffer space No packets are buffered and hence discarded when the users is not in the idle state. After the player did send a packet successfully, he has to wait until a new one is generated. This model is used in section 4.2 on page 60.

Finite buffer space The packets are buffered. If all of the buffer size $\mathcal{B}$ is used, the newly generated packets are discarded. If the buffer is empty, the user does not send any more packets. This model is analysed in section 1.2.3.

Infinite buffer space We assume, every player has an sufficiently large buffer and generates enough packet so he has data he wants to transmit at every time instance. He does send with probability q, regardless of whether the packet to be transmitted is a newly generated one or a packet that collided in a previous attempt. This model is the most frequently used throughout the paper.

Figure 1.3 illustrates the binomial model when using infinite buffer space.

1.2.3 Buffer Model

Description

In this section we want to examine the impact of a buffer of size $\mathcal{B}$ which was used in the previous section 1.2.2. In contrast to many papers and books (for example [vM06] on page 220) we don't restrict the calculations to the case $\mathcal{B} = 1$, but try to find

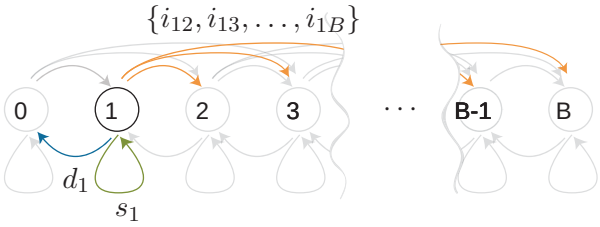


Figure 1.4. Illustration of the Markov chain used in section 1.2.3. There are three transition types possible: i,d and s. Of the latter two only one exists per state, denoted by d_S and s_S respectively (S being the origin of the transition). On the other hand multiple transitions of type i are possible for each state, denoted by i_{ST} (T representing the target state of transition).

a Markov chain that approximates the general case $\mathcal{B} > 1$.

We assume the following points:

Packet arrival rate: Each participant generates packages, that he likes to transfer over a shared channel. The arrival of newly generated packets is modeled by a Poisson process with rate λ.

Buffer space: Each user has a buffer of length $\mathcal{B}$ in which he can store packets that wait to be transmitted. The packet, which should be transferred at the moment, occupies one buffer space as well. As soon as the buffer is filled up, all packets arriving afterward are discarded.

The newly generated packets arrive *before* the users transmits packets. This fact matters when the buffer is empty or filled up (see for example equations (1.4b) and (1.4c)).

Protocol: All participants use the slotted ALOHA-protocol, so in every time slot they send with probability q - apart from the case of course, no packet is available.

The average number of packets generated per time slot, thus the rate of the Poisson process, is equal to λ. The random variable A counts the number of generated packets per time slot and is distributed as follows:

$$\Pr[A = k] = \frac{\lambda^k}{k!} e^{-\lambda} \tag{1.1}$$

Since each player behaves the same way we focus on player number 1 in the following. From the remaining $N - 1$ users only the information how many players want to send a packet is needed. We denote this number by $N_{\text{active}}^{(n)}$, where n stands for the current time slot. Hence, when user 1 sends after n time slots, the probability his packet does not collide is equal to:

$$\begin{aligned} \Pr\left[\text{no collision} \middle| \text{player 1 sending}\right]^{(n)} &= (1-q)^{N_{\text{active}}} \\ &=: \hat{P}_{suc}^{(n)} \end{aligned} \tag{1.2}$$

To determine the number N_{active} we need to work out a Markov chain as explained in the following section.

Markov chain

The number of occupied buffer space S is interpreted as states of a Markov chain. The chain therefore consists of $\mathcal{B} + 1$ states, state 0 representing the circumstance no packets are present and state $\mathcal{B}$ representing a completely filled buffer. Figure 1.4 shows the resulting Markov chain [GS03].

In the course of the following sections we give the transition probabilities of the Markov chain.

Decrease in the number of occupied buffer space

The transition d_S means, the occupied buffer space decreases, which only happens when no packet is newly generated while the users is successful in transferring a packet. This number depends on time and is given by the following product:

$$d_S^{(n)} = \Pr[A = 0] \cdot q\hat{P}_{suc}^{(n)} \quad \forall\, 0 < S < \mathcal{B}$$

with (1.1) we get

$$= e^{-\lambda} \cdot q\hat{P}_{suc}^{(n)} \qquad \forall\, 0 < S < \mathcal{B} \tag{1.3a}$$

When the buffer is filled up, it doesn't matter how many packets are generated, since they are all discarded. So:

$$d_{\mathcal{B}}^{(n)} = q\hat{P}_{suc}^{(n)} \tag{1.3b}$$

Since the number of stored packets cannot be come negative we have:

$$d_0^{(n)} = 0 \tag{1.3c}$$

Steady occupied buffer space When one of the following things happens, the buffer content does neither decrease nor increase:

1. Player 1 does neither send nor generate a packet.
2. Player 1 does send, but his packet collides. No packet is generated.
3. Player 1 does generate exactly one packet and transfer it successfully.

Therefore the probability the occupied buffer space S does not change after n time slots with probability (the numbers correspond to the items of the list above):

$$\begin{aligned} s_S^{(n)} &= \underbrace{(1-q)e^{-\lambda}}_{(1)} + \underbrace{q(1-\hat{P}_{suc}^{(n)})e^{-\lambda}}_{(2)} + \\ &\quad \underbrace{q\hat{P}_{suc}^{(n)}\lambda e^{-\lambda}}_{(3)} \qquad \forall\, 0 < S < \mathcal{B} \\ &= e^{-\lambda}\big(1 + q\hat{P}_{suc}^{(n)}(\lambda - 1)\big) \qquad \forall\, 0 < S < \mathcal{B} \end{aligned} \tag{1.4a}$$

If $S = 0$ (the buffer is empty) only the cases 1 and 3 can happen and hence:

$$s_0^{(n)} = e^{-\lambda}(1 + \lambda \cdot q\hat{P}_{suc}^{(n)}) \tag{1.4b}$$

If $S = B$ the buffer is already filled and it does not matter how many packets are generated since they are all discarded. The only condition is, that the user does not send a packet successfully:

$$s_{\mathcal{B}}^{(n)} = 1 - q\hat{P}_{suc}^{(n)} \tag{1.4c}$$

Increase in the number of occupied buffer space The transition i_{ST} with $T > S$ and $T < \mathcal{B}$ means, $T - S$ packets are added to the buffer. This happens when either $T - S$ packets are generated and no packet transmitted successfully (first addend in the following formula) or $T - S + 1$ packets are generated and one is transmitted successfully (second addend):

$$\begin{aligned} i_{ST}^{(n)} &= \big(1 - q\hat{P}_{suc}^{(n)}\big)\frac{\lambda^{T-S}}{(T-S)!}e^{-\lambda} \\ &\quad + q\hat{P}_{suc}^{(n)}\frac{\lambda^{T-S+1}}{(T-S+1)!}e^{-\lambda} \\ &= \frac{\lambda^{T-S}}{(T-S)!}e^{-\lambda} \cdot \\ &\quad \left(1 + q\hat{P}_{suc}^{(n)}\left(\frac{\lambda}{T-S+1} - 1\right)\right) \end{aligned} \tag{1.5a}$$

The transition $i_{S\mathcal{B}}$ means, at least $\mathcal{B} - S$ packets are generated and none of them is transmitted successfully. Hence:

$$i_{S\mathcal{B}}^{(n)} = (1 - qP_{suc}^{(n)}) \sum_{k=\mathcal{B}-S}^{\infty} \frac{\lambda^k}{k!}e^{-\lambda} \tag{1.5b}$$

The transition matrix after n time slots therefore is:

$$\mathsf{P}_B^{(n)} = \begin{pmatrix} s_0^{(n)} & i_{01}^{(n)} & \cdots & i_{0B}^{(n)} \\ d_1^{(n)} & s_1^{(n)} & \ddots & \vdots \\ & \ddots & \ddots & i_{(B-1)B}^{(n)} \\ & & d_B^{(n)} & s_B^{(n)} \end{pmatrix} \tag{1.6}$$

where the entries are given by (1.3), (1.4) and (1.5) respectively.

Using the starting distribution $\mathsf{u} = (1, 0, \ldots, 0)$ we can calculate the probability distribution of the occupied buffer space after n time slots as follows:

$$\mathsf{u}^{(n)} = \mathsf{u}\mathsf{P}_B^{(1)}\mathsf{P}_B^{(2)} \cdots \mathsf{P}_B^{(n)} \tag{1.7}$$

The expected number of $N-1$ users who want to transmit a packet now is approximated by the prob-

ability, the buffer is not empty times $N-1$:

$$N_{\text{active}}^{(n)} = (N-1)\mathrm{u} \cdot \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{1.8}$$

The effective sending probability q_{eff}

As we will show in section 1.3.1 on page 17, N players can transmit at most $\left(1-\frac{1}{N}\right)^{N-1}$ packets. For this reason if we have

$$\lambda > \frac{1}{N}\left(1-\frac{1}{N}\right)^{N-1} \tag{1.9}$$

the users generate more packets than they can send. The buffers will fill up and every player sends all the time with probability q after a sufficiently large time.

REMARK: Even when the arrival rate exceed the throughput, there is a chance the buffer is emptied completely. But the bigger the λ and $\mathcal{B}$, the smaller the probability (see table 1.1). For the sake of simplicity we will ignore these small chances and assume, as already mentioned, the player constantly wants to send. ◇

$\mathcal{B}$	λ/TP_{max}	Probability
20	1.05	1.81 %
50	1.05	0.127 %
10	1.1	0.007 %
20	2.6	$1 \cdot 10^{-13}$ %

Table 1.1. Probability the buffer is empty considering long term behavior when 10 users with q = 0.1 compete. λ/TP_{max} stands for the ratio of the arrival rate and the maximum throughput.

On the other hand, when less packets arrive than the participants could send, the users are not busy all the time trying to send packets. Figure 1.5 illustrates this circumstance: At a rate given by λ, new packets arrive. The participant thereafter tries to send this packet with probability q until he succeeds. This is expected to happen before a new

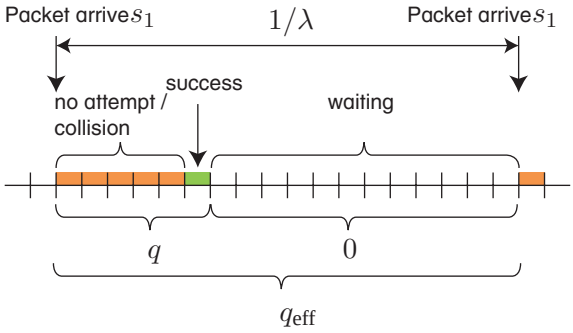


Figure 1.5. Illustrates the calculation of the effective sending probability q_{eff}. New packets arrive at intervals of average length $1/\lambda$, while it takes lesser time to send a packet. The remaining time, the player spends waiting for new packets. The ratio of time sending to the packet arrival interval determines the effective sending rate of the player.

packet arrives and he has to wait for this reason (which means sending with probability 0). In a rougher view, these two "stages" and their corresponding sending probabilities can be summarized to the effective sending probability, which is simply the expected sending probability given by:

$$q_{\text{eff}} = q\frac{1}{\frac{q\hat{P}_{suc}^{(\infty)}}{\frac{1}{\lambda}}} \qquad q\hat{P}_{suc}^{(\infty)} \geq \lambda$$

where the numerator corresponds to the expected number of time slots a participant needs to transmit a packet, while the denominator is equal to the interval till a new packet arrives.

$P_{suc}^{(\infty)}$ stands for the the probability of success after a sufficiently large amount of time when the system ceased to fluctuate. Using the effective sending probability q_{eff} the probability of success is equal to

$$P_{suc}^{(\infty)} = (1 - q_{\text{eff}})^{N-1} \tag{1.10}$$

which gives us the equation:

$$q_{\text{eff}} = \frac{\lambda}{(1-q_{\text{eff}})^{N-1}} \tag{1.11}$$

As we can see, the effective sending probability does not depend on the actual choice q! Since the following condition must be fulfilled:

$$\lambda \geq q \cdot \hat{P}_{suc}^{(\infty)}$$

the sending probability must however satisfy:

$$q \geq \frac{\lambda}{\hat{P}_{suc}^{(\infty)}}$$

Accuracy of the calculations

Now we want to analyze how good our approximation of the number of users N_{active} was. The equation (1.8) does not take into account the correlation between the users. Therefore, we simulated one million time slots with 10 users and a sending probability of $q = 0.12$. We then counted the number of users who concurrently tried to send a packets for every one of these times slots and compared the results to the values calculated as explained in section 1.2.3.

j	f_j^o	π_j^o	f_j^e	π_j^e
0	771951	(77.2%)	769942	(77%)
1	200443	(20%)	203948	(20.4%)
2	25348	(2.5%)	24310	(2.4%)
3	2128	(0.2%)	1717	(0.2%)
4	123	(0%)	80	(0%)
5	5	(0%)	3	(0%)
6	2	(0%)	0	(0%)
7	0	(0%)	0	(0%)
8	0	(0%)	0	(0%)
9	0	(0%)	0	(0%)
10	0	(0%)	0	(0%)

Table 1.2. Shows the number of users j concurrently sending. f_j^o (π_j^o) stand for the observed frequency (probabilty) and f_j^e (π_j^e) for the expected frequency (probability) calculated using the methods from section 1.2.3. The total number of users is 10, the sending probability $q = 0.12$ and the packet arrival rate $\lambda = 0.01$.

$\lambda = 0.01$: Using a packet arrival rate $\lambda = 0.01$, we got the simulation result compiled in the second column of table 1.2. As you can see 77.2 percent of the time none of the participants wanted to transmit data. The third column shows the (rounded) results of the calculations. Since we treated the users independently, the values are binomially distributed. We can now perform a goodness-of-fit test ([Bor99]) to test if the simulation results are as well binomially distributed with the same mean and variance (null hypothesis H_0) or if they differ significantly from our results (alternative hypothesis H_1).

The goodness-of-fit test is carried out by calculating the χ^2-value according to:

$$\chi^2 = \sum_{j=0}^{10} \frac{(f_j^o - f_j^e)^2}{f_j^e} \tag{1.12}$$

where f_j^o and f_j^e correspond to the observed and expected frequencies respectively. The degree of freedom is ten (df=10), since of the 11 possible numbers of users concurrently sending 10 can be freely chosen (the 11th one is given by the condition, the numbers must add up to 1'000'000). Using (1.12) we get

$$\chi^2 \approx 302$$

Comparing this to the significance level of 99% which is $\chi^2_{(10;99\%)} \approx 23$ we see, the null hypothesis (that the number of users sending is a random sample from a binominal distribution) has to be rejected. But since we used 1'000'000 samples, the difference can be very small. To estimate the order of magnitude we calculate the effect size given by ([Bor99])

$$\varepsilon = \sqrt{\sum \frac{(\pi_j^o - \pi_j^e)^2}{\pi_j^e}} \tag{1.13}$$

and we get $\varepsilon = 0.0174$ (very small effect). Comparing the observed success probability $P_{\text{suc,obs}}^{(\infty)} = 0.778$ with the one calculated according to (1.10) and (1.11) which is $P_{\text{suc,eff}}^{(\infty)} = 0.790$ we see an error of

$$\frac{P_{\text{suc,eff}}^{(\infty)} - P_{\text{suc,obs}}^{(\infty)}}{P_{\text{suc,obs}}^{(\infty)}} = 1.58\% \tag{1.14}$$

j	f_j^o	π_j^o	f_j^e	π_j^e
0	391840	(39.2%)	385897	(38.6%)
1	378427	(37.8%)	385508	(38.6%)
2	171376	(17.1%)	173304	(17.3%)
3	47858	(4.8%)	46168	(4.6%)
4	9109	(0.9%)	8071	(0.8%)
5	1284	(0.1%)	968	(0.1%)
6	94	(0%)	81	(0%)
7	11	(0%)	5	(0%)
8	1	(0%)	0	(0%)
9	0	(0%)	0	(0%)
10	0	(0%)	0	(0%)
10	0	(0%)	0	(0%)

Table 1.3. Shows the number of users j concurrently sending. f_j^o (π_j^o) stand for the observed frequency (probabilty) and f_j^e (π_j^e) for the expected frequency (probability) calculated using the methods from section 1.2.3. The total number of users is 10, the sending probability $q = 0.12$ and the packet arrival rate $\lambda = 0.038$.

j	f_j^o	π_j^o	f_j^e	π_j^e
0	278014	(27.8%)	278410	(27.8%)
1	380498	(38%)	379753	(38%)
2	232582	(23.3%)	233094	(23.3%)
3	84962	(8.5%)	84784	(8.5%)
4	20369	(2%)	20238	(2%)
5	3215	(0.3%)	3313	(0.3%)
6	333	(0%)	377	(0%)
7	26	(0%)	29	(0%)
8	1	(0%)	2	(0%)
9	0	(0%)	0	(0%)
10	0	(0%)	0	(0%)

Table 1.4. Shows the number of users j concurrently sending. f_j^o (π_j^o) stand for the observed frequency (probabilty) and f_j^e (π_j^e) for the expected frequency (probability) calculated using the methods from section 1.2.3. The total number of users is 10, the sending probability $q = 0.12$ and the packet arrival rate $\lambda = 0.1$.

Using $q_{\text{eff}} = 0.0258$ according to (1.11) and ignoring the correlations between the sending probabilities of the participants therefore is precise enough. The red lines in the four figures 1.6 (a) to (d) show the characteristics when using $\lambda = 0.01$ both simulated and calculated.

$\lambda = 0.038$: When increasing the arrival rate λ up to the maximum throughput the player can reach, we get the results compiled in table 1.3. The goodness-of-fit test yields:

$$\chi^2 \approx 557$$

And therefore the simulated results significantly differ from the binomial distribution ($\chi^2_{(10;99\%)} \approx 23$). As the higher value compared to the case $\lambda = 0.01$ already gives away, the effect size (according to (1.13)) is bigger than for the case $\lambda = 0.01$

$$\varepsilon = 0.0236$$

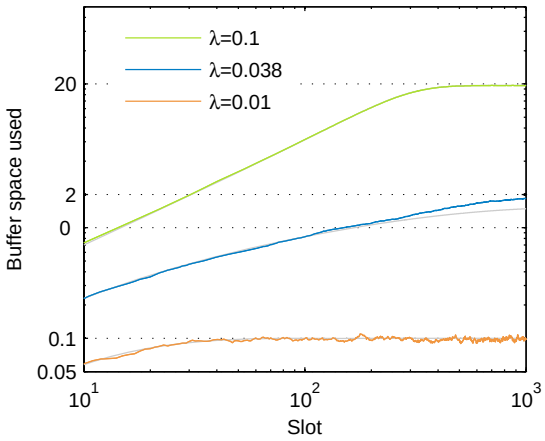
but still very small. The difference between the observed success probability $P^{(\infty)}_{\text{suc,obs}} = 0.417$ and the calculated result $P^{(\infty)}_{\text{suc,eff}} = 0.424$ is increased as well

$$\frac{P^{(\infty)}_{\text{suc,eff}} - P^{(\infty)}_{\text{suc,obs}}}{P^{(\infty)}_{\text{suc,obs}}} = 1.89\%$$

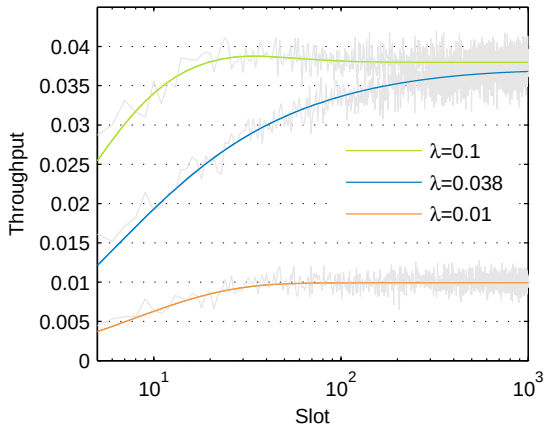
but still sufficiently small to yield meaningful results as the blue courses in the figures 1.6 (a) to (d) show.

$\lambda = 0.1$: Finally, we consider the case the arrival rate of packets exceeds the throughput of the participants and their buffers are filled up. In this case all participants send with probability $q_{\text{eff}} = q = 0.12$ at any time (the ratio of the arrival rate to the maximum throughput is approximately 2.6, as the last row in table 1.1 the buffer being empty considering long term behavior is very tiny). Therefore the sending behavior of the users cannot be correlated and the simulation results have to tally with the calculated results. As table 1.4 shows this is the case and since

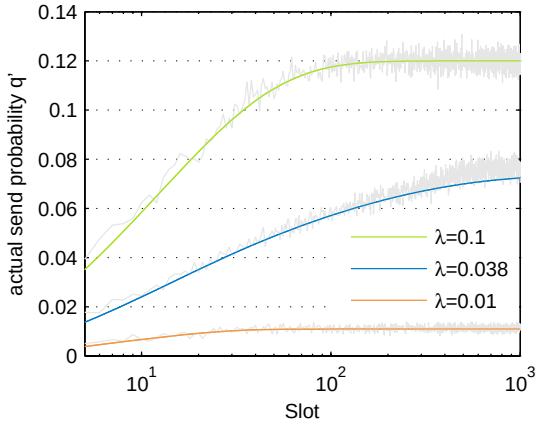
$$\chi^2 \approx 13$$



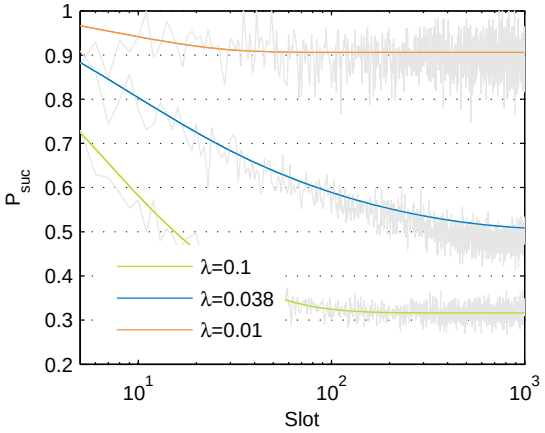
(a) Used buffer space: If more packet are generated than can be sent ($\lambda > q \cdot (1-q)^9 \approx 0.038$) by one user, the the buffer is filled up continuously. If equally many packet arrive as can be sent, approximately 2 packet slots are used on average in respective buffers. At an arrival rate of $\lambda = 0.01$ the buffer remains almost empty - only 0.1 slots are used on average. The calculated courses are very precise for the cases $\lambda = 0.1$ and $\lambda = 0.01$, while deviant for the case $\lambda = 0.038$.



(b) Throughput: If too many packet are generated, the user never lacks of material to send. The throughput therefore increases fast up to the maxium $q(1-q)^9$. If as many packets arrive as can be transmitted ($\lambda = 0.038$), the throughput as well strives for the maximum, but since entirely drained buffers can still arise, this maximum is not reached completely. If only $\lambda = 0.01$ packets per time slot arrive, they can all be sent easily and the throughput is equal to the arrival rate.



(c) q': At an arrival rate $\lambda = 0.1 > q$ the actual sending probability q' is equivalent to the real sending probability q. Interestingly an arrival rate approximately equal to the possible throughput $\lambda = 0.038$ results in a way lower actual sending probability of approximately 0.075. This can be explained by noting, that the throughput at $q = 0.12$ can also be reached by a sending probability smaller than 1. If too few packets arrive e.g. $\lambda = 0.01$, the chance of being successful when sending them, is very high and therefore $q' \approx \lambda$.



(d) P_{suc}: The probability of a being successful when sending a packet clearly is the bigger the smaller the arrival rate λ is. For λ greater than the maximum throughput, the probability approximates $(1-q)^9$.

Figure 1.6. Shows four interesting characteristics: buffer space used (a), throughput (b), actual sending probability (c) and the probability P_{suc} (d) for three different arrival rates λ. The gray lines represent the average of 1000 simulation runs. There are 10 users competing with a sending probability of $q = 0.12$..

compared to $\chi^2_{(10;99\%)} \approx 23$ the null hypothesis holds. The difference between the measured success probability $P^{(\infty)}_{\text{suc,obs}} = 0.315$ and the expected result $P^{(\infty)}_{\text{suc,eff}} = 0.316$:

$$\frac{P^{(\infty)}_{\text{suc,eff}} - P^{(\infty)}_{\text{suc,obs}}}{P^{(\infty)}_{\text{suc,obs}}} = 0.293\%$$

is purely by chance. The green lines in the figures 1.6 (a) to (d) show the characteristics when using $\lambda = 0.1$ both simulated and calculated.

Conclusion The statistical analysis carried out in section 1.2.3 showed, that the participants using buffers can be approximated using the Markov chain derived in section 1.2.3. When waiting a sufficiently large time, the buffers don't have do be considered anymore and it is enough to calculate the effective sending probability according to (1.11) on page 12 and thereafter proceed as in the model 1.2.2 described in section 1.2.2 on page 9. We will therefore not use the rather complex model derived in this section but the more simpler version 1.2.2 from section 1.2.2.

1.3 Efficiency of ALOHA

1.3.1 Throughput

As already mentioned, all packets are of the same length denoted by L. Let T be the time necessary to transmit a single packet. Given the bit rate B this time clearly is equal to $T = \frac{L}{B}$.

Pure ALOHA

In the time interval $(t, t+T)$ a successful transmission takes places only when:

1. Exactly one participant sends a packet with the starting time lying between t and $t+T$. Let this starting time be t' with $t \leq t' < T$.
2. The packet does collide, which means, no other users starts transmitting a packet within the period $(t'-T, t'+T)$.

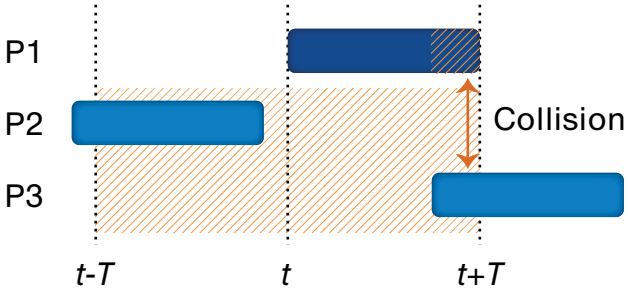


Figure 1.7. Illustration of the pure ALOHA protocol with three Players $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$. To ensure, that the packet of the first player sent at time instance t does not collide, the remaining players are not allowed to send within the red hatched time period of length $\mathbf{2T}$.

These facts are represented by figure 1.7.

The probability of exactly k of the total of N participants try to transmit a packet in a given time interval of length T is, using the first model, equal to:

$$\Pr[k, N] = \binom{N}{k} q^k (1-q)^{N-k} \tag{1.15}$$

And considering the second model:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \tag{1.16}$$

With the considerations carried out so far the throughput according to the first model can be calculated as follows:

$$\begin{aligned} \text{TP}_1 &= \Pr[0, N-1] \Pr[1, N] \\ &= (1-q)^{N-1} \cdot Nq(1-q)^{N-1} \\ &= Nq(1-q)^{2(N-1)} \end{aligned} \tag{1.17}$$

Or when using the second model:

$$\begin{aligned} \text{TP}_2 &= \Pr[0] \Pr[1] \\ &= e^{-G} \cdot Ge^{-G} \\ &= Ge^{-2G} \end{aligned} \tag{1.18}$$

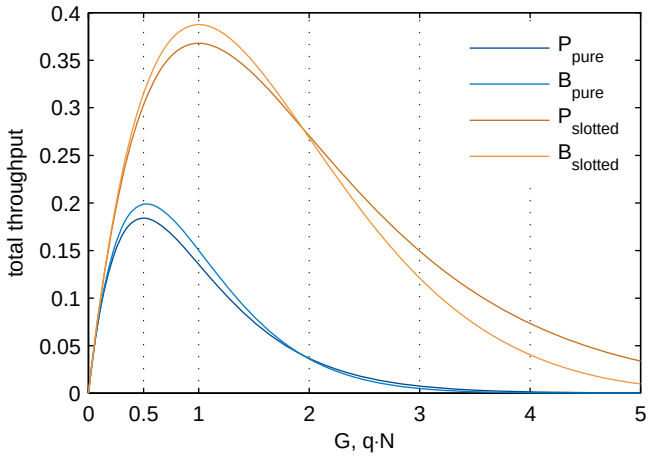


Figure 1.8. Total throughput of packets per frame time T for $N = 10$ participants. B_{pure} and $B_{slotted}$ show the throughput of pure and slotted ALOHA respectively according to the first model subject to the product $q \cdot N$, where q stands for the sending probability. P_{pure} and $P_{slotted}$ stand for the throughput after the second model in dependence of the load G

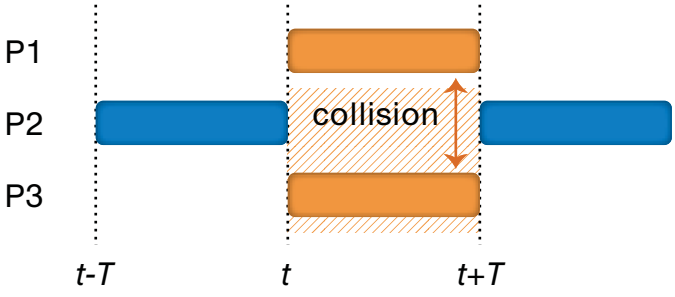


Figure 1.9. Example of slotted ALOHA with three users. In contrast to figure 1.7 the transmission can start at the beginning of given time slots. Therefore, the critical range (red hatched) within no other transmission are allowed, is cut in half.

Differentiating (1.17) leads to:

$$\frac{d\mathrm{TP}_1(q)}{dq} = N(1-q)^{2(N-1)-1} \cdot [1 - q(2N-1)]$$

The maximum throughput according to the first model is therefore achieved at the following sending probability:

$$q_{opt} = \frac{1}{2N-1} \tag{1.19}$$

On the other hand the following results from differentiating (1.18):

$$\frac{d\mathrm{TP}_2(G)}{dG} = e^{-2G}\,(1-2G)$$

And the throughput is maximized for $G = 0.5$ and is equal to $\mathrm{TP}_{2,\max} = 1/2e \approx 0.184$. The courses B_{pure} and P_{pure} in figure 1.8 show the throughput according to the first and second model respectively subject to the parameter Nq and G respectively when 10 users participate.

Slotted ALOHA

In contrast to the pure ALOHA-protocol described in the previous section, the users are not allowed to send anytime, but only at the beginning of a time slot of length T. This ensures a packet sent at time instance t' only collides, when another participant sends a packet within the period $(t, t+T)$ (An therefore starts the transmission at the point in time t' too.). Figure 1.9 shows these considerations.

The throughput according to the first model is:

$$\begin{aligned}\mathrm{TP}_1 &= \Pr[1, N] \\ &= Nq(1-q)^{N-1}\end{aligned} \tag{1.20}$$

The derivative of the function (1.20) is:

$$\frac{d\mathrm{TP}_1(q)}{dq} = N(1-q)^{N-2}(1-qN)$$

and the throughput is maximized for

$$q_{\mathrm{opt}} = 1/N \tag{1.21}$$

According to the second model we have:

$$\mathrm{TP}_2 = Ge^{-G}$$

And TP_2 differentiated vs G:

$$\frac{d\mathrm{TP}_2(G)}{dG} = e^{-G}(1-G)$$

The maximum throughput is therefore reached at

$$G_{opt} = 1 \tag{1.22}$$

and is of size $G = 1/e \approx 0.368$, twice as big as the maximum throughput in the pure ALOHA case. The lines $B_{slotted}$ and $P_{slotted}$ in figure 1.8 show the throughput according to the first and second model subject to the offered load and sending probability respectively when 10 users participate.

1.3.2 Delay

Pure ALOHA

Using the binomial model Let the random variable D denote the delay of a given packet. Since the probability, a particular Player transmits a packet in the timespan T, is equal to $P_{suc}^{(T)} = q(1-q)^{2(N-1)}$ according to (1.17). Therefore the probability he doesn't send or the transmission attempt fails within the period T is equal to $\overline{P}_{suc}^{(T)} = 1 - P_{suc}^{(T)}$. Since this probability is time-independent, we can generalize the formula to:

$$\begin{aligned}\overline{P}_{suc}^{(t)} &= \left(\underbrace{1 - q(1-q)^{2(N-1)}}_{\xi}\right)^{\frac{t}{T}} \\ &= \xi^{\frac{t}{T}} \\ &= P(D \geq t) \\ &= 1 - F(t) = F_c(t)\end{aligned}$$

Where $F(\cdot)$ ($F_c(\cdot)$) stand for the (complementary) cumulative distribution of the delay D. Since we know $F(\cdot)$ we can easily calculate the probability density function $f(\cdot)$ of the random variable D by differentiation:

$$\begin{aligned}F(t) &= 1 - \xi^{\frac{t}{T}} && \forall t \geq 0 \\ \Rightarrow f(t) &= F'(t) \\ &= -\frac{1}{T} \cdot \ln(\xi) \xi^{\frac{t}{T}} && \forall t \geq 0 \end{aligned} \tag{1.23}$$

N	pure	slotted
5	22.6	11.2
10	49.8	24.8
20	104.2	52
50	267.2	133.6
100	539.1	269.5

Table 1.5. Shows the mean delay in the binomial model when using the optimal sending probability q. As you can see, the delay values in the pure version are roughly twice as big as for the slotted cases.

Finally the expected delay $\mathrm{E}[D]$ is:

$$\mathrm{E}[D] = \int_0^\infty t f(t)\, dt$$

Using (1.23) this leads to

$$\begin{aligned}&= -\frac{T}{\ln \xi} \\ &= -\frac{T}{\ln\left(1 - q(1-q)^{2(N-1)}\right)}\end{aligned} \tag{1.24}$$

The delay is minimized for the same value q that maximizes the throughput since the logarithmic function is strict monotonic increasing and its argument is equal to (1.17) except for a constant factor. When using the optimal q the delay-values compiled in second column of the table 1.5 result.

Using the Poisson model At every transmission attempt the chances of succeeding is equal to e^{-2G} where we used (1.16). Therefore the number of attempts A until a packet is transferred (including the successful attempt) is given as follows:

$$\Pr[A = a] = \left(1 - e^{-2G}\right)^{a-1} \cdot e^{-2G}$$

The expected value of attempts $\mathrm{E}[A]$ can be computed as the sum:

$$\begin{aligned}\mathrm{E}[A] &= \sum_{a=1}^{\infty} a \cdot \Pr[A = a] \\ &= e^{-2G} \sum_{a=1}^{\infty} a \left(1 - e^{-2G}\right)^{a-1} \\ &= e^{2G}\end{aligned}$$

Every packet that takes a transmission attempts, is delayed $a - 1$ times by the amount of time the player waits. Let the mean waiting time be $\mathcal{W}$, then the expected delay is equal to:

$$\mathrm{E}[D] = \mathcal{W} \cdot (\mathrm{E}[A] - 1) \tag{1.25}$$

$$= \mathcal{W}\left(e^{2G} - 1\right) \tag{1.26}$$

(Where we used the linearity property of the expected value operator). For the throughput-optimal case $G = 0.5$ the expected delay is approximately $1.7\mathcal{W}$.

Slotted ALOHA

Using the binomial model For every time slot a specific user sends a packet successfully with probability $P_{suc} = q(1-q)^{N-1}$ (see equation (1.20)). Therefore the probability a packet is delayed for d time slots before sent is equal to:

$$\begin{aligned}\Pr[D = d] &= \sum_{d=0}^{\infty} d(1 - P_{suc})^d P_{suc} \\ &= \frac{1 - P_{suc}}{P_{suc}} \\ &= \frac{1 - q(1-q)^{N-1}}{q(1-q)^{N-1}}\end{aligned} \tag{1.27}$$

As one can see easily, the delay (1.27) is minimized for the same choice $q = \frac{1}{N}$ that maximizes the throughput as well. Some exemplary values are listed in the third column of table 1.5.

Using the Poisson model The mean delay can be derived analogously we did in section 1.3.2. The only difference lies in the probability of sending a packet successfully, which now is equal to e^{-G}. Therefore the expected delay is equal to:

$$\mathrm{E}[D] = \mathcal{W}\left(e^{G} - 1\right) \tag{1.28}$$

Where $\mathcal{W}$ is the mean waiting time. Using the optimal offered load $G = 1$ leads to the same expected delay in terms of waiting time $\mathrm{E}[D] \approx 1.7\mathcal{W}$.

1.4 Group of unfair users

1.4.1 Procedure

In this section we assume, that not all N participants behave the same way, but are rather split into a group of N_u unfair users and a group of N_f fair user, where $N_f + N_u = N$. We examine the slotted ALOHA-protocol and assume, the total number of users N is known.

The fair users try to maximize the total throughput and therefore send with the optimal sending probability according to (1.21)

$$q_f = \frac{1}{N}$$

or according to (1.22) generate the load

$$G_f = \frac{N_f}{N}$$

If all players stuck to this sending probability, the total throughput (1.20) would be maximized. However, the unfair participants now try to reach the following points:

1. The throughput of their group should be maximized.
2. Each unfair participant should achieve the same throughput on average.
3. Agreements between the unfair players don't take place and no departures from the ALOHA-protocol such as increased packet sizes or not sending within the given time slots are allowed. (We assume that such measures would be too easy to detect, even if the users don't reckon specifically with thieves.)

In this section, we want to examine how the throughput can be optimized if all unfair participants send with constant probability. In sections 2 and 3 on pages 27 and 33 respectively, a more flexible protocol is examined. The second demanded point can simply be fulfilled through symmetry, which means all unfair participants send with the same probability q_u. The throughput for unfair and fair players respectively now corresponds to the likelihood exactly one respectively fair or unfair players send. Therefore we have:

$$\mathrm{TP}_{1,f} = (1-q_u)^{N_u} \cdot N_f q_f (1-q_u)^{N_f-1}$$
$$\mathrm{TP}_{1,u} = (1-q_f)^{N_f} \cdot N_u q_u (1-q_u)^{N_u-1}$$

Or considering the second model:

$$\mathrm{TP}_{2,f} = e^{-G_u} \cdot G_f e^{-G_f}$$
$$\mathrm{TP}_{2,u} = e^{-G_f} \cdot G_u e^{-G_u}$$

Differentiating $\mathrm{TP}_{1,u}$ vs q_u results in:

$$\frac{d\mathrm{TP}_{1,u}(q_u)}{dq_u} = (1-q_f)^{N_f} N_u (1-q_u)^{N_u-2} \cdot (1-q_u N_u)$$

The optimal sending probability of unfair users therefor is equal to:

$$q_{opt,u} = \frac{1}{N_u} \tag{1.29}$$

And in the same way when differentiating $\mathrm{TP}_{2,u}$ vs G_u

$$\frac{d\mathrm{TP}_{2,u}(G_u)}{dG_u} = e^{-G_f-G_u}(G_u - 1)$$

we get the optimal offered load $G_{opt,u}$ being equal to 1. In both models the optimal behaviour of unfair participants does not depend on either the number or sending probability of fair users, but only on the size of unfair players.

Figure 1.10 shows the throughput for unfair and fair users in a mixed group of $N = 50$ participants. The throughput is normalized by dividing the obtained values by the throughput that would result from the symmetric case $q_f = q_u = 1/N$.

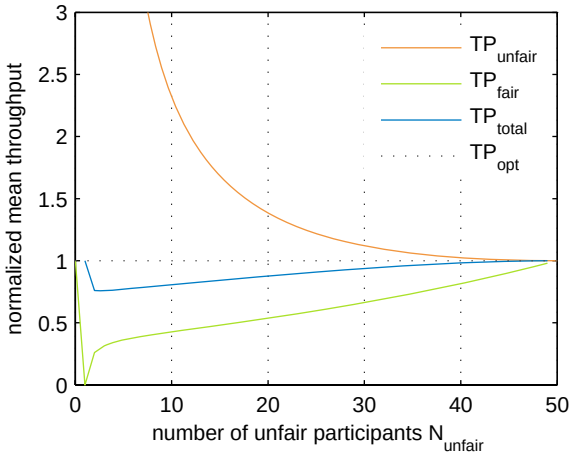


Figure 1.10. Shows the normalized throughput (subject to the all-fair-case) for fair ($\mathrm{TP}_{\mathrm{fair}}$) and unfair ($\mathrm{TP}_{\mathrm{unfair}}$) participants as well as the mean achieved throughput ($\mathrm{TP}_{\mathrm{total}}$) in dependence of the number of unfair users when the total number of users N is held constant at 50. As a reference the throughput ($\mathrm{TP}_{\mathrm{opt}}$) resulting from the case all users being fair is plotted as a dotted line.

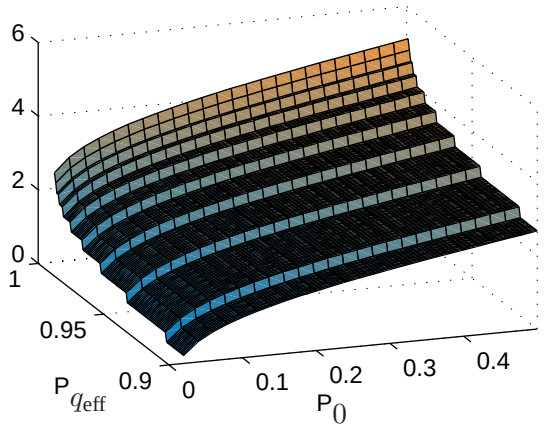


Figure 1.11. Factor, by which the 10 unfair participants can increase their sending probability such that they are detected by the remaining 90 users only with probability P_c, when they only sound the alarm they are certain to at least to P_d that they don't initiate an unwanted alarm signal. The measuring time is set to 100 time slots.

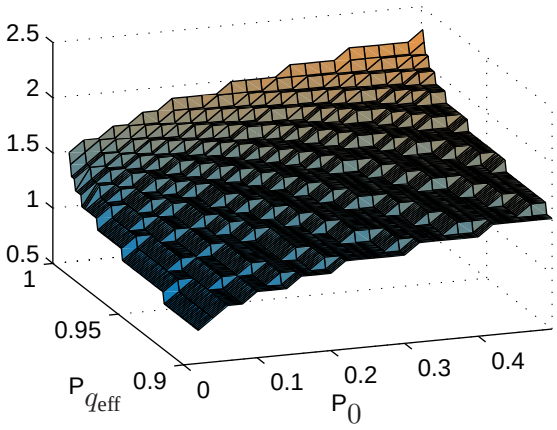


Figure 1.12. Shows the factor, by which the 10 unfair user can increase their sending probability without being caught by the 90 remaining fair users with probability $1 - P_c$. The fair must be certain P_d certain in order to sound the alarm. The measuring time is equal to 1000.

1.4.2 Detection of unfair users

In this section we examine how the group of unfair participants can be uncovered. We assume, the total number of participants is known to to all users. Therefore every user can calculate the expected number of collision within a given number of time slots. Since the unfair participants send with an elevated probability, an increased number of collision is measured on average. With the help of elementary statistical calculations, we can now give the probability all users are fair and send with probability $q = 1/N$ for any given number of collisions. Let P_d be the certainty that unfair users exists, that must be fulfilled so that the users sound the alarm due to the elevated number of collisions.

The unfair participants of course know these calculations and can therefore choose their elevated sending probability q_u in a way that they are caught at most with likelihood P_c [Bor99].

The figure 1.11 and 1.12 show, how large the unfair players can set their sending probability for a given P_d and P_c at most. The longer the measuring time, the smaller the unfair participants have to choose q_u.

Figure 1.13 shows the bounds for q_u for different P_d-P_c-constallation subject to the measuring time in slots.

1.5 Estimating the number of participants

As we have seen in section 1.3 on page 16, the number of users has to be known by each participant in order to maximize the throughput and minimizing the delay. Since the ALOHA-protocol has no setup phase or agreements between players, in this section, the task should be examined, how the number of players can be estimated, using the measured collision probability. We focus on the slotted ALOHA-protocol, but the result for unslotted ALOHA can simply be derived.

1.5.1 All users conforming to the rules

In this section we assume, that at first all N participants conform to the rules and therefore act the

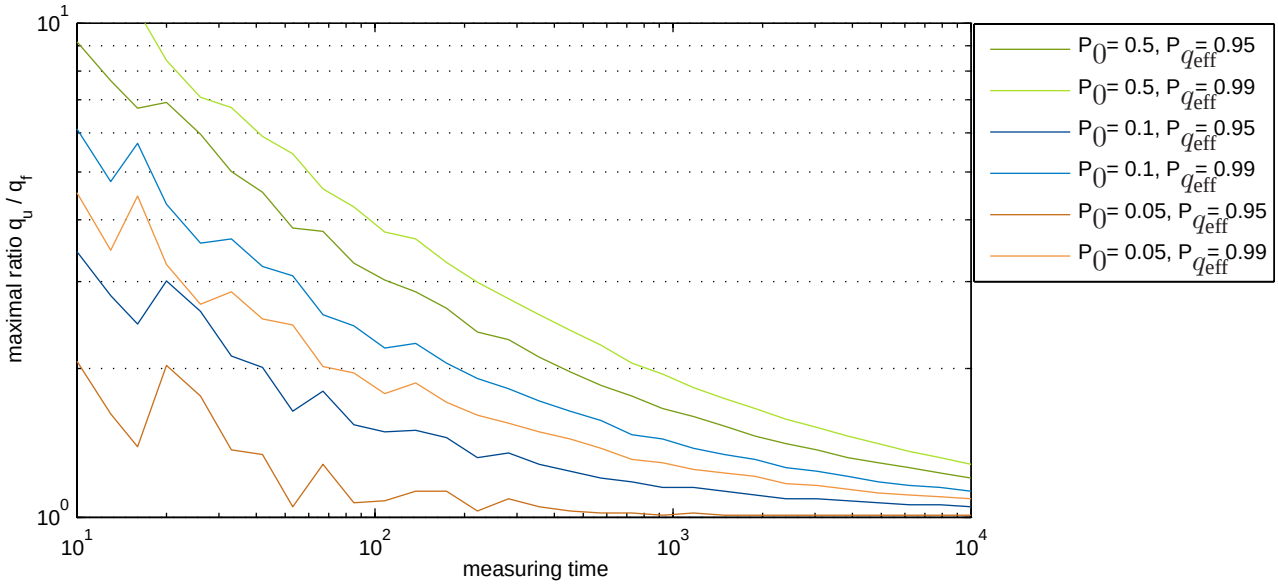


Figure 1.13. Maximum possible sending probability of the 10 unfair users without being uncoverd by the remaining 90 users. The more time slots are used to count collisions, the less aggressive the unfair users can behave.

same way and send with the same likelihood q. The following collision probability P_{col} results:

$$P_{col} = 1 - (1-q)^N - Nq(1-q)^{N-1} \tag{1.30}$$

Or according to the Poisson model:

$$= 1 - e^{-G} - Ge^{-G} \qquad N \gg 1 \tag{1.31}$$

Let C_S be the random variable standing for the number of slots out of N slots, in which collisions occured. C_S is binomially distributed and has the following probability density function:

$$P(C = c) = \binom{S}{c} \cdot P_{col}^c \cdot (1 - P_{col})^{S-c} \tag{1.32}$$

If the number of users N is large enough and the skew is not to big, the distribution can be approximated by the normal distribution:

$$W^S_{P_{col}}(c) \approx \frac{1}{\sqrt{2\pi S P_{col}(1-P_{col})}} \cdot \exp\left(-\frac{(c - SP_{col})^2}{2SP_{col}(1-P_{col})}\right)$$

REMARK: A possible rule of thumb to decide whether the approximation is valid is given by the following tests:

$$N \cdot P_{col} \overset{?}{>} 5$$
$$N \cdot (1 - P_{col}) \overset{?}{>} 5$$

◇

The task is to estimate the unobserved parameter N on the basis of the observed number of collisions C.

$$N \mapsto f(C|N)$$

Using (1.32) and (1.30) we have:

$$f(C|N) = \binom{S}{c} \cdot \left(1 - (1-q)^N - Nq(1-q)^{N-1}\right)^c \cdot \left((1-q)^N - Nq(1-q)^{N-1}\right)^{S-c}$$

The maximum likelihood estimate of N is

[Pro01]:

$$\hat{N}_{\mathrm{ML}}(x) = \arg\max_N f(C|N)$$

This estimate can be obtained by solving the following equation for $\bar{N}$:

$$\hat{P}_{col} = \left(1 - (1-q)^{\bar{N}} - \bar{N}q(1-q)^{\bar{N}-1}\right) \quad (1.33)$$

or when $N \gg 1$ and $G = Nq$:

$$\approx \left(1 - e^{-\bar{N}q} - \bar{N}qe^{-\bar{N}q}\right) \quad (1.34)$$

Where the estimated collision probability $\hat{P}_{col}$ can be obtained from the observed number of collision C by the relation:

$$\hat{P}_{col} = \frac{C}{S}$$

Using equation (1.34) we have:

$$\bar{N} \approx -\frac{1}{q}\,\mathrm{Lam}_{-1}\big((\hat{P_{col}} - 1)\cdot e^{-1}\big) - 1 \quad (1.35)$$

Where Lam_{-1} denotes the $k = -1$ branch of the Lambert W-function.

REMARK: Since (1.34) is a strictly monotonically increasing function, only one positive and real $\bar{N}$ satisfies equation (1.35) and the principal value of the Lambert W-function results in a negative, hence impossible, estimate of $\bar{N}$. ◇

Since N must be an integer, the maximum likelihood estimate is obtained by rounding $\bar{N}$ to the next number.

The exact equation (1.33) can be inverted as well, but the result is rather complex:

$$\bar{N} = \frac{q\,\mathrm{Lam}_{-1}\left(\frac{-(\hat{P}_{col}-1)\cdot\ln(1-q)}{q}\exp\left(\frac{\ln(1-q)}{q}\right)\right)}{q\ln(1-q)} - \frac{(1-q)\ln(1-q)}{q\ln(1-q)}$$

From the equation

$$C = S\big(1 - (1-q)^{\bar{N}} - \bar{N}q(1-q)^{\bar{N}-1}\big) := g(N)$$

the probability density distribution for $\bar{N}$ follows:

$$W_{\bar{N}}(\bar{n}) = W^S_{P_{col}}(g(\bar{n}))\cdot\left|\frac{dg(\bar{n})}{d\bar{n}}\right| \quad (1.36)$$

Where

$$\frac{dg(\bar{n})}{d\bar{n}} = -S\Big((1-q)^{\bar{n}-1}\big(\ln(1-q)(1-q-\bar{n})+q\big)\Big) \geq 0 \quad (1.37)$$

As we now know the most likely number of N, we can calculate the optimal send probability $\hat{q}$ based on this estimate according to (1.21) on page 17:

$$\hat{q} = \frac{1}{\hat{N}} = h(\hat{N})$$

REMARK: In fact, we could directly use the rational and therefore impossible estimate $\bar{N}$ to calculate the send probability q without having rounded the value. ◇

The estimate of q is distributed as follows:

$$W_{\hat{Q}}(\hat{q}) = W_{\hat{N}}(h(\hat{q}))\left|\frac{dh^{-1}(\hat{q})}{dq}\right|$$

Where

$$\frac{dh^{-1}(\hat{q})}{d\hat{q}} = -\frac{1}{\hat{q}^2}$$

The distribution of $\hat{q}$ follows from (1.36) and (1.37):

$$W_{\hat{Q}}(\hat{q}) = W^S_{P_{col}}\left(-\frac{1}{\hat{q}^2}\right)\cdot\frac{1}{\hat{q}^2}\cdot\left.\frac{dg(\hat{n})}{d\hat{n}}\right|_{\hat{n}=-\frac{1}{\hat{q}^2}}$$

Figure 1.14 shows, which throughput on average results when using the optimal q according to the estimated number of N when using four different measuring loads G_0 and send probabilities $q_0 = G_0/N$. As you can see, the choice $G_0 \approx 2$ yields the best estimations. If the measuring load is by the factor ten to small in comparision with the throughput maximizing load $G = 1$, it takes

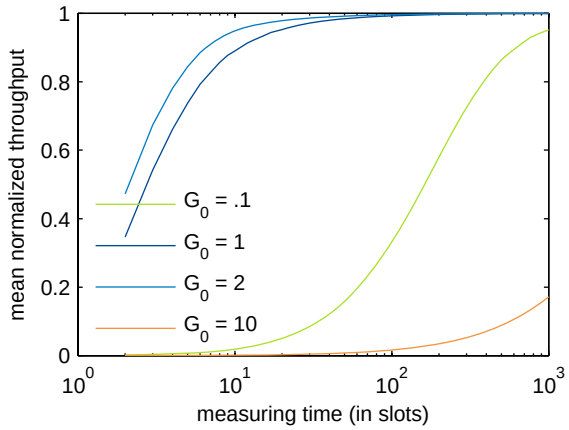


Figure 1.14. Mean normalized throughput for different send probabilities q_0 and loads $G_0 = q_0 \cdot N$ respectively depending of the number of slots . The number of users N - all being fair - was set to 50. When using the optimal measuring test load $G_0 \approx 2$ a good estimation - and hence good throughput - can be reached after only 50 time slots.

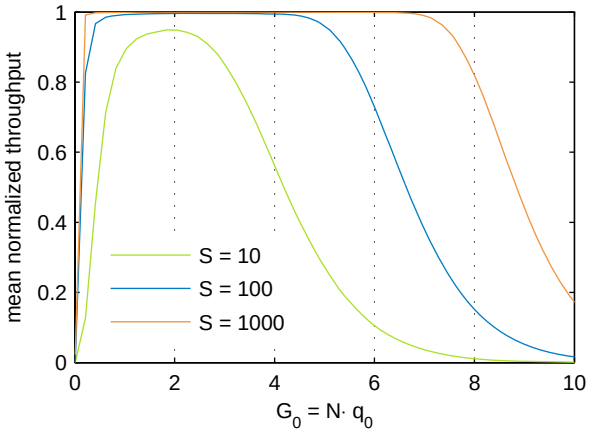


Figure 1.15. Mean normalized throughput subject to the test load $G_0 = q_0 \cdot N$ ($N = 50$) for three different measuring times S. The best estimations are obtained when $1 \leq G_0 \leq 4$ holds.

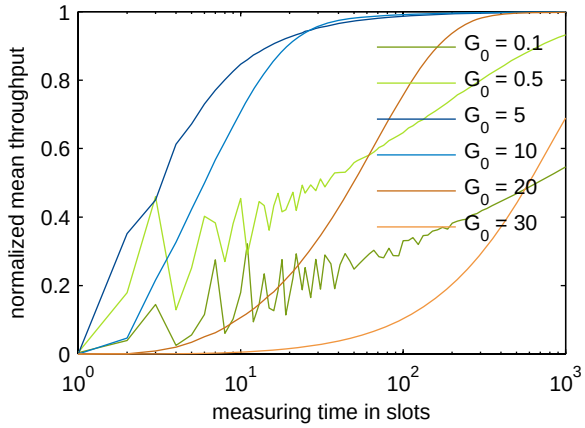


Figure 1.16. Estimating the number of unfair participants $(N_u = 50)$ when the number of total users $(N = 200)$ is known.

1000 time slots to reach a reliable estimation. On the other hand, with a measuring load to big by the same factor ($G = 10$) even after 1000 time slots the resulting estimation $\hat{q}$ is very inaccurate in most cases.

Figure 1.15 shows, within which bounds the test load has to lie in order to obtain a reliable estimation $\hat{q}$. These bounds are shown for different measuring times S.

1.5.2 A group of unfair users

In this subsection we assume, the total number of N is known. Of those, N_u participants are unfair. In order not to betray themselves, they cannot be recognized as unfair players and therefore, the number N_u is unknown to the unfair as well as the fair players.

The task therefore is, to estimate the number of unfair users with the same procedure we used in section 1.5.1. The estimation can thereafter be used by the unfair participants to set their sending probability according to (1.29) (page 20)in order to maximize their throughput.

Figure 1.16 shows how well the estimation fits in dependence of the measuring time for different measuring loads $G_{0,u}$ when $N_u = 50$ and $N_f =$

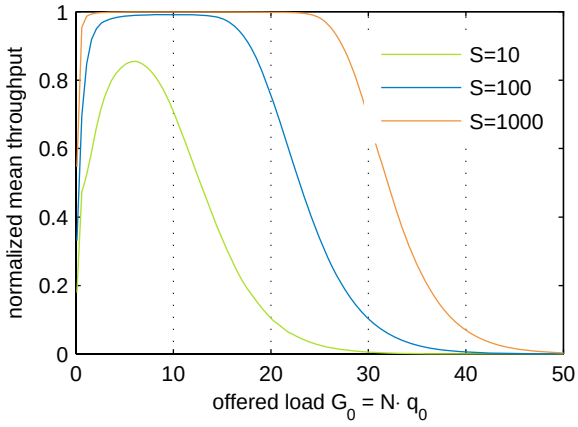


Figure 1.17. Mean normalized throughput against the test load G_0 for three different measuring times.

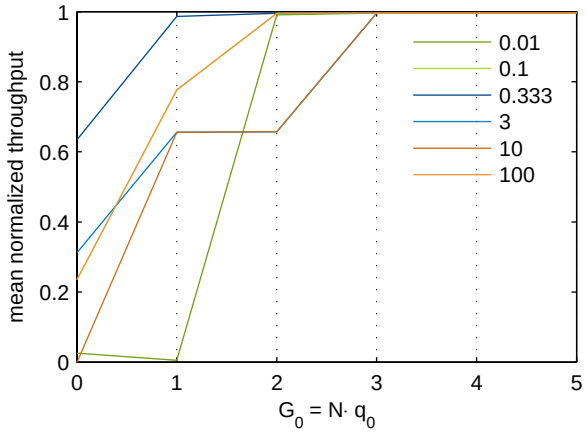


Figure 1.18. Shows the mean normalized throughput, that is reached when repeating the estimation procedure. The values at 0 correspond to the throughput obtained when using the test load G_0, the values at 1 corresponds to the throughput obtained by using the first estimation of q and so on. The measuring time S was set to 100, the number of users N is 100.

150. At the load $G_{0,u} \approx 5$ a reliable estimation is reached in the shortest time. With this load, the total amount of packets per time slot $N_u q_u + N_f q_f$ is approximately equal to $5\frac{50}{200} + 150\frac{1}{200} = 2$ and therefore, like in the previous section, twice as large as the offered load in the optimal case.

Figure 1.17 shows the achieved normalized throughput for three different measuring times in dependence of the test load $G_{0,unfair}$.

1.5.3 Repeated estimation

The estimated optimal sending probability $\hat{q}$ from sections 1.5.1 and 1.5.2 can be used to transmit packets. By the way the number of collisions can be counted and used to obtain a new estimation of the number of users. This can be particularly useful, if we expect users to leave or join the protocol and the users may increase or have to decrease their sending rate. We restrict the calculations to case, the number of users stays the same and analyse, how much estimation enhances by iterating the measurments.

The ith estimation of q is distributed as follows:

$$W^i_{\hat{q}_0}(\hat{q}_i) = \int_0^1 \cdots \int_0^1 f_{\hat{q}_1}(\hat{q}_2) \cdots f_{\hat{q}_{i-1}}(\hat{q}_i) \cdot \hat{q}_0 \, d\hat{q}_1 \cdots \, d\hat{q}_n$$

Where $f_{\hat{q}_1}(\hat{q}_2)$ denotes the distribution of the second estimation of q based of the measuring results obtained when sending according to the first estimation of q.

Figure 1.18 shows, how the estimation enhances when using 100 time slots and the procedure is repeated.

Chapter 2

Adaptive sending probabilities

In this section another method should be examined, beside the estimation of the number of users 1.5, how the unfair users can increase their throughput altogether as much as possible. We assume that the total number of users N is known (at least to the fair participants), however not the number of the unfair players N_u.

2.1 Procedure

The fair participants always send with probability $q_f = \frac{1}{N}$, which means they aim for the optimal throughput of the group of all users. The unfair players, however, behave like follows: At first, they send with the same probability as the unfair users $q_u = q_f$ for the duration of X time slots and measure the achieved throughput TP_1. After this measuring time each unfair player increases his sending probability by α percent and transmit again for X time slots. Afterward, each unfair player i checks if the achieved throughput TP_2 is bigger than the throughput obtained with the first choice of $q_u^{(i)}$, the player again increases his send probability by the factor α. If, on the other hand, the throughput decreased, the player decreases his send probability by α percent. This procedure is repeated permanently. The goal of the procedures is to bring the values $q_u^{(i)}$ of the unfair players to the optimal values, which is according to (1.29) (page 20) equal to:

$$q_u^{(i)} = \frac{1}{N_u} \tag{2.1}$$

REMARK: Optimal means the throughput of the group of unfair players (the sum of the individual throughputs of unfair players) is maximized, and every unfair player achieves the same throughput. $\diamond$

2.2 Throughput reached

The figures 2.1 and 2.2 show the throughput the group of the unfair participants, the group of the fair users as well as both together respectively can achieve on an average in a specific run and averaged over 1000 runs. The throughput was normalized in a way that the value 1 corresponds to the throughput the considered user would reach when all users are sending with the optimal probability $q = 1/N$ according to equation (1.21). The maximal possible increase the 3 users can achieve is equal to:

$$\begin{aligned} f &= \frac{q_u(1-q_u)^{N_u-1}(1-q_f)^{N_f}}{q_f(1-q_f)^{N-1}} \\ &\approx 1.83 \end{aligned} \tag{2.2}$$

In practice, the unfair users reach a mean throughput around three times bigger than when behaving fair as you can see in figure 2.1 . The reasons for the discrepancy between (2.2) and the simulation results are clarified in the next section.

2.3 Cannibalism

2.3.1 Reasons

As figure 2.1 showed, the throughput of the unfair group can be improved heavily by the adaptive

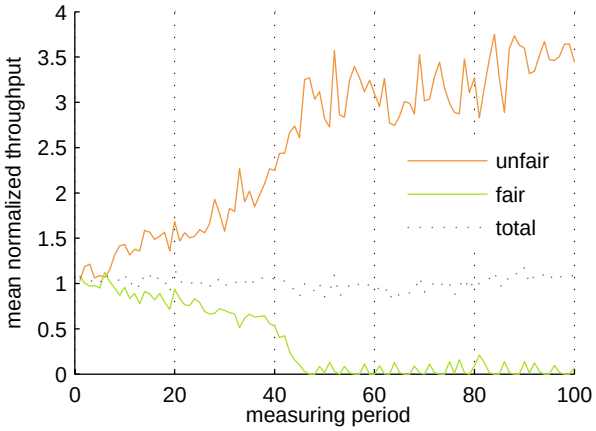


Figure 2.1. Shows the mean normalized throughput for 7 fair (TP_{fair}) and 3 unfair users (TP_{unfair}) as well as the mean total achieved throughput (TP_{total}). The simulation run was carried out with a measuring time of 8192 time slots and a increase/decrease parameter $\alpha = 10\%$. The average throughput of the unfair participants is increased roughly by the factor 3.5 after 100 measurement periods. The throughput of the unfair players on the other hand drops to zero.

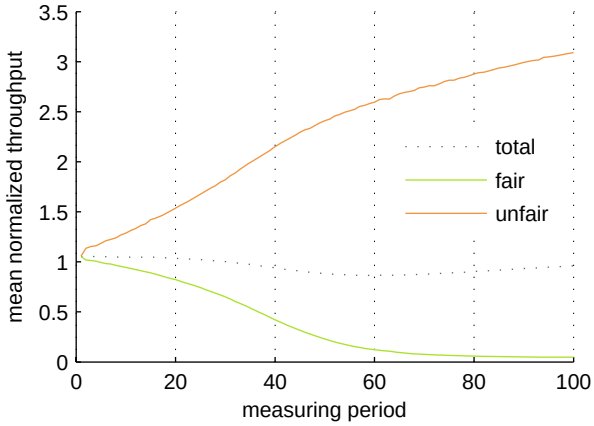


Figure 2.2. Illustrates the mean normalized throughput for 7 fair (TP_{fair}), 3 unfair users (TP_{unfair}) and in total (TP_{total}) respectively subject to the measuring period, each of which 8192 time slots long. All throughput values are calculated as the average of 1000 simulation runs. The parameter α is set to 10 %.

procedure explained in section 2.1. It was not examined however until now, whether this throughput is distributed *equally* among the unfair players as demanded in section 1.4.1. When examining the individual throughput courses of unfair players in figure 2.3 an astonishing fact is revealed: The three unfair players don't increase their sending probability in the same order of magnitude but rather exactly one of the unfair players constantly increases his probability up to 1 at the expense of the remaining participants, including his allied unfair players.

At the end, the two other unfair participants send with a probability, that corresponds approximately to sending probability of the fair players. Consequently, the unfair player who constantly sends and cannibalizes all others, achieves a throughput TP_{can}:

$$TP_{can} \approx (1 - q_f)^{N-1} \tag{2.3}$$

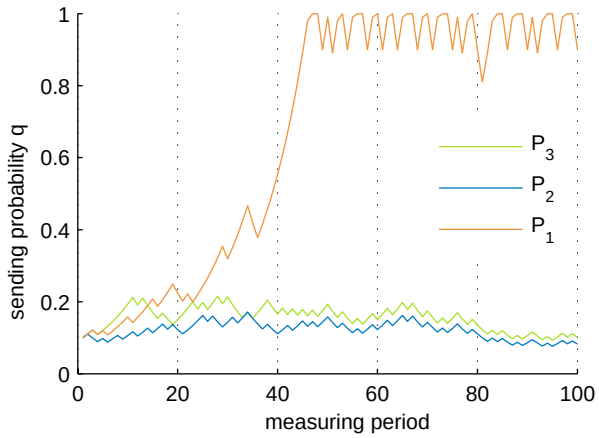


Figure 2.3. Shows the sending probabilities of the three unfair users in detail for a measuring time of 8192 slots and $N_f = 7$ fair participants. The parameter α was set to 0.1. As you can see, on of the unfair users sends constantly at the end, while the remaining unfair users remain at a moderate sending probability level.

while the remaining player never succeeds in transmitting a packet:

$$\mathrm{TP}_{rest} = 0 \tag{2.4}$$

The throughput of the cannibalizing users is increased by the factor N compared to the case, he and all competitors behave fairly. When incrementing and decrementing the sending probability by α percent, this cannibalizing behavior occurs for *every* choice of α.

2.3.2 Detection

Since the unfair users are bound by an agreement to each other, even the cannibalizing participant, who actually profits, has an interest in changing his behavior. To do so, however, he has to be able to realize that he is doing so. Or at least, the remaining unfair users should notice that they are victims, too.

It will be shown with the help of a specific setting, that either of the requests can be satisfied.

2.3.3 Countermeasure

When incrementing and decrementing the sending probability by α percent, this leads to cannibalism as seen in section 2.3. One obvious thing to prevent cannibalism or at least make it more difficult for a participant to increase his sending probability while forcing back the sending probability of the remaining members of the unfair group is the following: The bigger the q becomes, the smaller the increase steps are. Therefore unfair users who tend to be restrained by one or several participants can catch up with the more aggressive users. This kind of rubber band effect can be achieved with various methods. For example, the increase or decrease of q in terms of the percentage of q represented by $\alpha(q)$ could be chosen as follows:

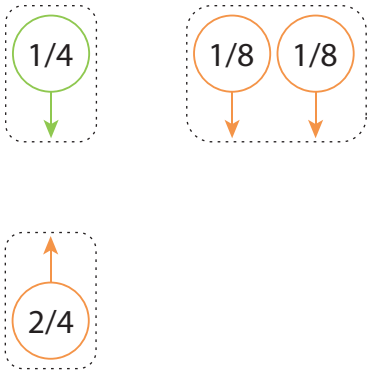
$$\alpha_{\mathrm{v1}}(q) = \beta \cdot (1 - q) \tag{2.5a}$$

or

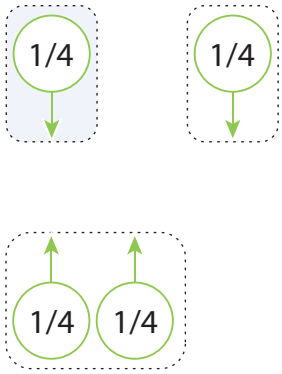
$$\alpha_{\mathrm{v1}}(q) = \beta \cdot \big(- \log(q) \big) \tag{2.5b}$$

Where β is a constant. Both choices (2.5a) and (2.5b) prevent cannibalism, but as one can see in figure 2.5 another, even worse, problem arises: The unfair users all increase their sending probabilities nearly the same way, but they don't stop in doing so when the group throughput is reached, but continue until they all send constantly and the throughput drops to zero.

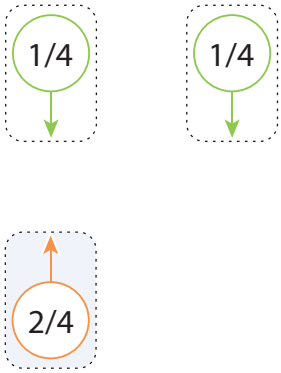
Figure 2.6 shows the course of the throughput of the fair group, the unfair group and both together respectively. After a initial increase up to the theoretical maxiumum, the throughput of unfair users begins to drop to zero.



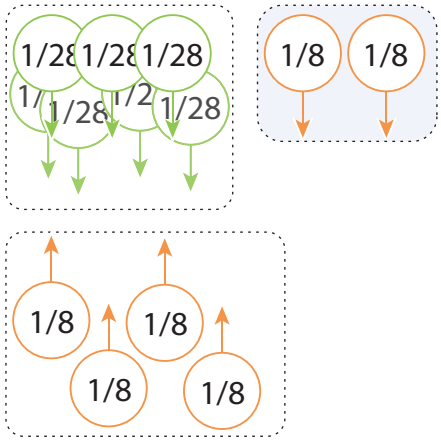
(a) Shows the reality: Four users take part in the protocol, three unfair (orange) ones and one fair player (green). One of the unfair users has increased his sending probability too far (meaning over the level $1/N_u = 1/3$ up to 50%). The two remaining unfair users are suppressed to a sending probability 1/8 and therefore smaller than 1/3, in fact even smaller than the value of the fair user.



(b) Shows the view of the fair player (highlighted): The packets of the two suppressed unfair users are regarded as coming from only one user, exactly vice-versa the cannibalizing participants is considered as two users. By this interpretation, the fair player thinks there are three additional fair users taking part in the protocol all sending with probability 1/4.



(c) This figure shows the view of the user, that cannibalizes the remaining unfair players. He can simply interpret these two users as being one fair participants. With this interpretation his sending probability of 0.5 is not too big, he should on the contrary increase the value up to 1 (what in fact happens in reality).



(d) The repressed users finally can also interpret their fate in a way, that they have the feeling to send at the cost of the fair participants and sending as often as the unfair companions. The single fair player is divided up to 7 participants, which send with a probability that is much smaller than their own. The cannibalizing unfair user is interpreted as four players who send with the probability 1/8 and therefore at the same rate.

Figure 2.4. Since only the packet load is recognized and not the emitting source, the reality (a) can be interpreted in different ways. So all three types of users (framed) can interpret the reality in a way that they don't determine any anomaly.

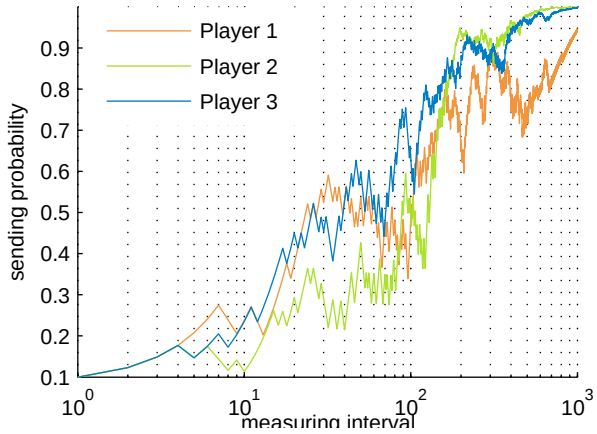


Figure 2.5. This figure shows the courses of the sending probabilities $q^{(i)}$ for three unfair players, when the adaptation α is made dependent on the current value of q. 7 fair users, who send steadily with 10 percent, were involved additionally. The measuring intervals had a length of 1000 time slots. As one sees, none of the unfair players is put at a disadvantage anymore. However, the values of q rise all up to 1, when this happens, none of the users can transfer a packet successfully.

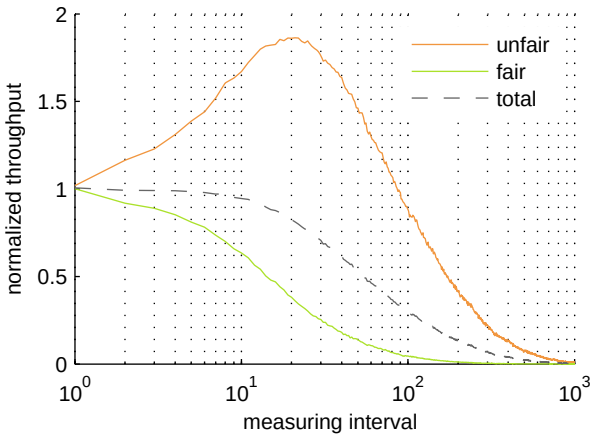


Figure 2.6. Shows the throughput of the group of unfair players ($N_u = 3$) as well as the throughput of fair users ($N_u = 3$) when using a sending probability adaption that depends on the actual values of q. The unfair players reach the desired point where their throughput is maximized, but unfortunately don't stop increasing the sending probability up to 1 and the throughput values drop to zero.

Chapter 3

ALOHA with induced regularity

It was assumed until now that the sending probability of the participants is time- and delay-independant. In this section it shall be shown how the throughput of all players can be increased with the help of a delay dependent send probability. In doing so a single player only needs to be informed when one of his packets collides, more detailed information or even arrangements are not necessary.

Aloha protocols at which the sending probability depends on the delay of the packages (and possibly on a state parameter) are called *variable-q-ALOHA* in the following.

In this chapter, the model 1.2.2 from the section 1.2.2 on page 9 should be used.

3.1 Version 1: Transient

In this section a simple version of a variable-q-ALOHA protocol when the the send probability only depends on packet-delays is given. The protocol is carried out - except for the variable sending probability - like the real ALOHA-protocol.

3.1.1 Procedure

Idea

By a well directed choice of the sending probabilities it should be tried to bring a certain regularity into the ALOHA-protocol, which increases the throughput and reduces the delay.

In the this section, we firstly go into the calculation of the throughput and the delay, the latter being rather laborious. The accuracy of the results derived is determined by comparing them with simulation results.

In the section 3.3.1 on page 43 we will give in-depth information how long it takes until we get the desired regularity and how big the delay and throughput are in this state.

In the section section 3.5 on page 54 finally it is examined how unfair users can apply the protocol when competing with fair users who behave according to the conventional ALOHA–protocol described in section 1.3 on page 16 and what the effects on delay and throughput are for both the fair and unfair players.

Name of the protocol

As we will see in section 3.1.2 it is advisable to differentiate the users by the delay of their current packet into two states. Since every participant can switch from the one state to the other and back, we call the protocol *transient* opposed to the protocol described in section 3.2 on page 39.

Protocol description

The only thing a player has to do, is counting the number time slots his current packet is delayed and sending in every time slots according to the sending probability q_{d+1} where d stands for the delay. All participants share the same sending probability vector q given in section 3.1.2.

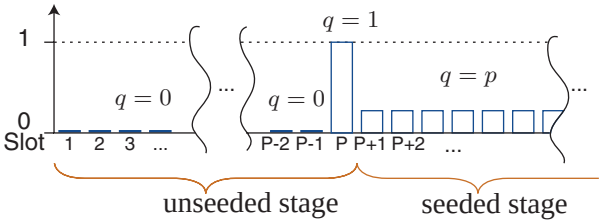


Figure 3.1. This illustration show the participants behaviour analysed in subsection 3.1. During P-1 time slots after last successful transmission the user keeps quiet and sends in the Pth slot a 100 per cent. A player who is in this seeded stage is called *seeded*. As soon as the delay becomes greater than P, the player sends with a constant probability p. These participants are referred to as unseeded

.

3.1.2 Mathematical model

Choice of the sending probability

q_i denotes the likelihood a participant sends in the ith time slot after a successful transmission. These q_i are given as follows (see illustration 3.1):

$$q_i = \begin{cases} 0 & i < P \\ 1 & i = P \\ p & i > P \end{cases} \tag{3.1}$$

When the protocol was started so, nobody would hazard a transmission attempt during $P-1$ time slots and send in the Pth slot guaranteed (what leads to a collision, except for the degenerate case of only a participant). It is assumed for this reason in the following, that at initialization every player starts at the state $i = P+1$.

Dividing into two states

In order to calculate the throughput and the delay exactly we would need to take track of the delay of every user in order to determine is sending probability at any given time instance. But since this is not feasible, we only want to know, how many users have a delay greater than P and therefore send with constant probability p, and how many users have a delay smaller or equal to P and for this reason send if and only if the delay is equal to P.

The states $i \leq P$ are assigned to the seeded stage, the remaining to the unseeded stage (see figure 3.1. Users who are in the seeded stage (and therefore have sent at the most P time slots ago for the last time), are labeled *seeded players*, those in the unseeded stage *unseeded players*.

To calculate the performance, particularly the throughput, of the protocol all users (N in number) are divided up into two groups: Firstly a group of seeded players (with N_s participants) and secondly a group of unseeded players (with N_u participants). Of course $N = N_s + N_u$.

REMARK: Please note that N_u has a different meaning in this chapter than it had in chapters 2 and 1, where it represented the number of unfair players. ◇

Markov chain

States We interpret the number of seeeded users as states of a Markov chain now. So the state 0 corresponds to no seeded players, 1 corresponds to one seeded player and so on up to the state N at which all players are seeded. Altogether the chain contains $N+1$ states. Every time a new time slot starts, a transition in the Markov chain takes place (see figure 3.2).

REMARK: If more users are participating than the length of the period, which means $N > P$, only up to P users can be seeded at the same time. Hence the markov chain only has $P+1$ states, as shown in figure 3.3. ◇

Transition probabilities Since a player changes from the unseeded to the seeded stage (see figure 3.1) if and only if he has transferred a package successfully, N_s can increase by one per transition at the most. A seeded player switches to the unseeded group if and only if he is at the state $i = P$ and one of the unseeded player transmits a packet. Since only one player can be in the state $i = P$ (no two players can have a sent a packet successfuly P time

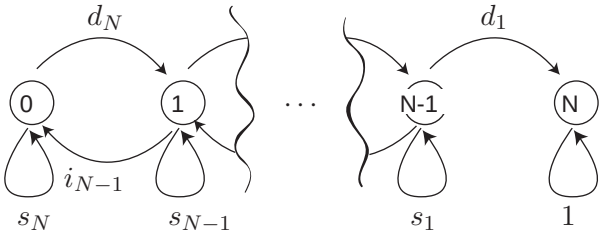


Figure 3.2. Markov chain examined in section 3.1. The states 0 to N stand for the corresponding number of seeded users. Three transitions types are possible, named s,d and i.

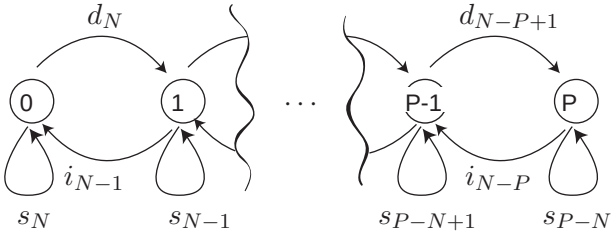


Figure 3.3. Markov chain examined in section 3.1 for the case $N > P$. The states 0 to N stand for the corresponding number of seeded users. Three transitions types are possible, named s,d and i. Please note that in contrast to the case $\mathbf{N \leq P}$ (figure 3.2) the state N (and as a consequence the whole Markov chain) is not absorbing.

slots ago) at the same time, N_s can decrease by the amount 1 at the most. If nobody sends or a collision occurs among seeded players, the groupsizes N_s and N_s remain constant. So all the transitions shown in illustration 3.2 are possible (For the case $N > P$ please consult figure 3.3).

We mark the probability that a state change $X_{n+1} = N - N_u + 1, X_n = N - N_u$ takes place by d_{N_u}. As mentioned this is the case if no seeded player sends (probability $1 - \frac{N_s}{N}$) and at the same time exactly one unseeded players sends (probability $N_u\, p\, (1-p)^{N_u - 1}$). Therefore:

$$d_{N_u} = \frac{(P - N + N_u)}{P} N_u\, p\, (1-p)^{N_u - 1} \quad (3.2)$$

Let i_{N_u} be the probability of a removal of a seeded player and going frm the state $N - N_u$ to the state $N - N_u - 1$. It is given by the product of the probabilities that a seeded player sends ($\frac{N_s}{N}$) and at least one unseeded player sends ($1 - (1-p)^{N_u}$). Hence:

$$i_{N_u} = \frac{(N - N_u)}{P} \left(1 - (1-p)^{N_u}\right) \quad (3.3)$$

The last possible transition from $N - N_u$ to $N - N_u$ is denoted by s_{N_u} and has the value $1 - d_{N_u} - i_{N_u}$:

$$s_{N_u} = \frac{(P - N + N_u)}{P} \left(1 - N_u\, p\, (1-p)^{N_u - 1}\right) + \frac{(N - N_u)}{P} (1-p)^{N_u} \quad (3.4)$$

We can take the first summand in (3.4) as the likelihood that no seeded player sends while no unseeded player transmits a packet successfully and the second summand as the probability a seeded players sends successfully.

REMARK: The equations (3.4), (3.2) and (3.3) are only valid for $N_u > N - P$. ◇

Let P be the transition matrix with elements p_{ij}, the probability of going from state i to state j in a single-step. For the case $N \leq P$ this matrix is given by:

$$\mathsf{P} = \begin{pmatrix} s_N & d_N & & & \\ i_{N-1} & s_{N-1} & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & i_1 & s_1 & d_1 \\ & & & & 1 \end{pmatrix} \quad (3.5)$$

Where s, i and d are defined by (3.4), (3.3) and (3.2) respectively (the zeros are not shown). For the case $N > P$ the transition matrix is:

$$\mathsf{P} = \begin{pmatrix} s_N & d_N & & \\ i_{N-1} & s_{N-1} & \ddots & \\ & \ddots & \ddots & d_{N-P+1} \\ & & i_{N-P} & s_{N-P} \end{pmatrix} \quad (3.6)$$

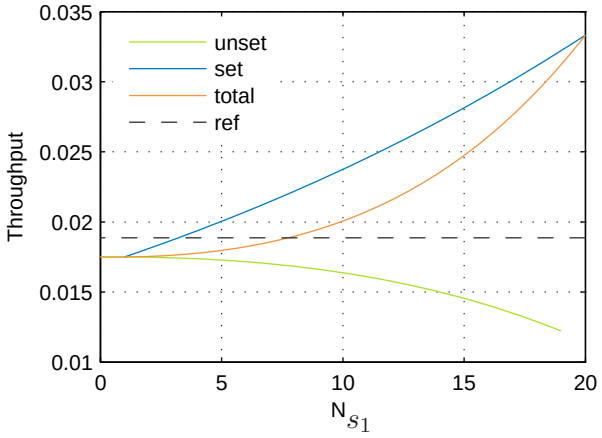


Figure 3.4. Throughput of seeded and unseeded players as well as the the total throughput subject to the number of unseeded players. The throughput of classical slotted ALOHA is shown as dashed line. The total number of player is 20, the period length P equals 30. In second stage users send with probability $p = \frac{1}{P}$.

At the beginning of the protocol none of the players is seeded, this circumstance is represented with the probability vector u which represents the starting distribution:

$$\mathsf{u} = (1, 0, \ldots, 0) \tag{3.7}$$

Then the probability vector after n slots is

$$\mathsf{u}^{(n)} = \mathsf{u} \cdot \mathsf{P}^n \tag{3.8}$$

3.1.3 Throughput

To be able to calculate the throughput after a certain time, at first the throughput which is reached at a given number of unseeded players is calculated. The throughput $\mathrm{TP}^{\star}_{N_u}$ corresponds to the probability a package is transferred successfully, either a package of a seeded player (first addend in the following equation) or a package of a unseeded player (second addend):

$$\begin{aligned}\mathrm{TP}^{\star}_{N_u} &= \frac{(N - N_u)}{P}(1-p)^{N_u} \\ &\quad + \frac{(P - N + N_u)}{P} N_u\, p\, (1-p)^{N_u - 1}\end{aligned} \tag{3.9}$$

Illustration 3.4 shows the throughput of seeded and unseeded players as well as the total throughput vs. the number of seeded players (N_s). As a comparison the throughput which would be reached by the classic ALOHA–protocol is plotted. As one sees, the total throughput gains for increasing number of N_s, however the unseeded players suffer losses.

We can combine the elements of (3.9) in the vector $\breve{\mathsf{TP}}$ (note the descending order of the $\mathrm{TP}^{\star}_{N_u}$).

$$\breve{\mathsf{TP}} = (\mathrm{TP}^{\star}_{N_u}, \mathrm{TP}^{\star}_{N_u - 1}, \ldots, \mathrm{TP}^{\star}_{\max(N-P,0)}) \tag{3.10}$$

The expected throughput after n slots is given by:

$$\begin{aligned}\mathrm{TP} &= \mathsf{u}^{(n)} \cdot \breve{\mathsf{TP}} \\ &= \sum_{i=0}^{\min(P,N)} u_i^{(n)} \breve{\mathrm{TP}}_i\end{aligned} \tag{3.11}$$

REMARK: The functions $\max(N - P, 0)$ and $\min(P, N)$ in (3.9) and (3.11) respectively generalize the formulas to any choice of N either $N \leq P$ or $N > P$. ◇

Where $\mathsf{u}^{(n)}$ and $\breve{\mathsf{TP}}$ are given by (3.8) and (3.10) respectively. Figure 3.5 gives the average throughput of ten thousand simulated results as well as the course of the calculated function values according to (3.11). As one sees, the periodical behaviour (particularly strong for early times) is not grasped by the function. The relative error therefor decreases with the time, as you can see in figure 3.6, since the sawtooth behavior is weakening. The send probability of unseeded players p equals $1/P$.

3.1.4 Delay

Mathematical model

Effortful calculations are necessary to calculate the average delay of the packets. Bookkeeping has to

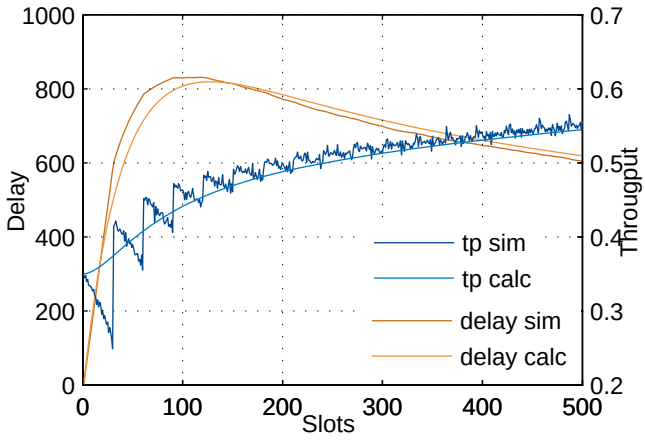


Figure 3.5. Progression of the throughput and delay for 10'000 simulated runs with 20 participants and a period length 30. The calculated courses of the functions (3.11) and (3.24) are plotted as well.

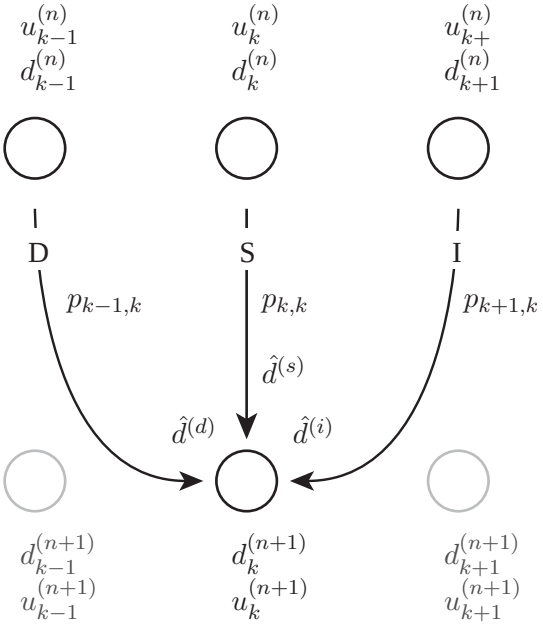


Figure 3.7. Illustrates the calculation of the upcoming delay values (see section 3.1.4). U stands for the state distribution vector, d for the delay, p for the transition probabilities. The three possible transitions are called D, S and I respectively.

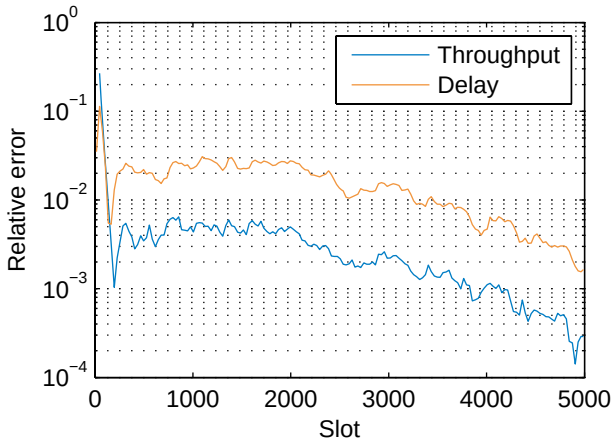


Figure 3.6. Average error of the calculated results (3.11) and (3.24) relative to ten thousand simulation runs with 20 players, P=30 and p=1/P. The error values within each period were combined to compensate the periodical behaviour of the throughput.

be done for every point in time n what delay for every possible number of unseeded players is expected. This information is stored in the vector $\mathsf{d}^{(n)}$ with the elements $d_i^{(n)}$ which are equal to the chance of beeing in the state i after n timeslots multiplied by the delay all unseeded players have accumulated in this case. If, for example, the protocol after 3 time slots is 30% in the first state with an expected delay of 20 and to 70% in the second state with a delay of 10, the vector $\mathsf{d}^{(3)}$ would be $(6, 7)$. The initial delay obviously is $\mathsf{d}^{(0)} = (0, \ldots, 0)$.

Each state $i^{(n+1)}$, except for $0^{(n+1)}$ and $N^{(n+1)}$, has three predecessors states: $(i-1)^{(n+1)}$, $i^{(n+1)}$ and $(i+1)^{(n+1)}$. The corresponding transitions are denoted by D,S and I respectively (see figure 3.7). The new delay value $\mathsf{d}_k^{(n+1)}$ therefore consists of the delay elements of the three paths, denoted by $\hat{d}^{(d)}$,$\hat{d}^{(s)}$ and $\hat{d}^{(i)}$. Each of these amounts can be divided up in three components:

$\boldsymbol{d_o}$ Delay contribution which is based on the delay already accumulated.

$\boldsymbol{d_+}$ Delay contribution which is added in the current time slot.

$\boldsymbol{d_-}$ Delay contribution which is discontinued in the current time slot.

These three delay shares are derived in the following section 3.1.4 and combined to the total delay in the section after next (section 3.1.4).

Delay shares

Calculation of $\mathbf{d_o}$ To determine the delay contribution d_o which belongs to a given path, the old contribution is multiplied by the corresponding transition probability p_{ij}. The contributions therefore are (see figure 3.7):

$$\begin{aligned}\hat{d}_o^{(d)} &= p_{k-1,k} \cdot d_{k-1}^{(N)} \\ \hat{d}_o^{(s)} &= p_{k,k} \cdot d_k^{(N)} \\ \hat{d}_o^{(i)} &= p_{k+1,k} \cdot d_{k+1}^{(N)}\end{aligned}$$

This can be summarized to the following equation:

$$\mathsf{d_o}^{(N+1)} = \mathsf{d}^{(N)} \cdot \mathsf{P} \tag{3.12}$$

Calculation of $\mathbf{d_+}$ The condition u_k corresponds to a unseeded number of players of $N_u = N-k$. At the transitions D and S $N-k$ unseeded users did not send successfully, thereby the delay increases by the same amount $N-k$. By multiplying these values by the corresponding state probabilities $u^{(n)}$ and transition probabilities $p_{\cdot,k}$ following contributions arise:

$$\begin{aligned}\hat{d}_+^{(d)} &= u_{k-1}^{(n)} \cdot p_{k-1,k} \cdot (N-k) \\ \hat{d}_+^{(s)} &= u_k^{(n)} \cdot p_{k,k} \cdot (N-k)\end{aligned}$$

The transition I corresponds to $N-k-1$ unseeded participant whose transmission attempts fails and one additional seeded player whose transmission fails as well. For every unseeded participant the delay increases by one time slot. Up to the current time, the seeded players delay was not represented in the accounting of the delay of the unseeded players, hence he brings in his complete accumulated delay - which is alwaly P. Therefore:

$$\hat{d}_+^{(i)} = u_{k+1}^{(n)} \cdot p_{k+1,k} \cdot (N-k-1+P)$$

This can be summerized by the vector representation as follows:

$$\mathsf{d_+}^{(N+1)} = \mathsf{u}^{(n)} \cdot (\mathsf{D_+} \star \mathsf{P}) \tag{3.13}$$

where $\star$ denotes the element-wise matrix multiplication:

$$\mathsf{C} = \mathsf{A} \star \mathsf{B} \Rightarrow c_{ij} = a_{ij} \cdot b_{ij} \tag{3.14}$$

The matrix $\mathsf{D_+}$ is according to $d_+^{(d)}, d_+^{(o)}, d_+^{(i)}$ given by:

$$\mathsf{D_+} = \begin{pmatrix} N & N-1 & & & \\ N-1+P & N-1 & N-2 & & \\ & N-2+P & \ddots & \ddots & \\ & & \ddots & \ddots & 0 \\ & & & P & 0 \end{pmatrix} \tag{3.15}$$

Calculation of $\mathbf{d_-}$ Along the paths I and S no packet is transmitted successfully. Hence, the delay does not decrease:

$$\begin{aligned}\hat{d}_-^{(P)} &= 0 \\ \hat{d}_-^{(i)} &= 0\end{aligned}$$

In the case of the transition D exactly one unseeded player transmits successfully a packet and his delay has to be substracted. This values varies, since it is unknown for how long the lucky player had to wait for the successful transmissions. Because every player had the same chance of succeeding, the expected discontinued delay is $d_{k-1}^{(N)}/(N-k+1)$. Hence:

$$\mathrm{E}\left[\hat{d}_{-}^{(d)}\right] = p_{k+1,k} \cdot \frac{d_{k-1}^{(N)}}{(N-k+1)}$$

So in the vectorial representation we have:

$$\mathsf{d}_{-}{}^{(n+1)} = \mathsf{d}_{-}{}^{(n)} \cdot (\mathsf{D}_{-} \star \mathsf{P}) \tag{3.16}$$

where $\star$ denotes the element-wise matrix multiplication. The matrix D_{-} is given by the elements $d_{-}^{(d)}$:

$$\mathsf{D}_{-} = \begin{pmatrix} 0 & \frac{1}{N} & & & \\ & 0 & \frac{1}{N-1} & & \\ & & 0 & \ddots & \\ & & & 0 & \frac{1}{1} \\ & & & & 0 \end{pmatrix} \tag{3.17}$$

Total delay

The equations (3.12), (3.13) and (3.16) yield:

$$\begin{aligned} \mathsf{d}^{(n+1)} = \mathsf{d}^{(n)} \cdot \mathsf{P} + \mathsf{u}^{(n)} \cdot (\mathsf{D}_{+} \star \mathsf{P}) - \\ \mathsf{d}^{(n)} \cdot (\mathsf{D}_{-} \star \mathsf{P}) \end{aligned} \tag{3.18}$$

where P stands for the matrix of transition probabilities according to (3.5) and $\mathsf{u}^{(n)}$ corresponds to the state distribution after n steps. D_{+} and D_{-} are given by (3.15) and (3.17) respectively. $\star$ denotes the element-wise matrix multiplication in accordance with (3.14).

The total delay finally consists of the delay of the unseeded as well as the seeded participants. The former is given by the sum of the elements of d:

$$\begin{aligned} D_{\text{unseeded players}} &= \left|\mathsf{d}^{(\mathsf{N})}\right|_1 \\ &= \sum_{i=0}^{N} d_i^{(N)} \end{aligned} \tag{3.19}$$

$|\mathsf{v}|_1$ stands for the 1-norm of vector v. The later, referring to the delay of the seeded players, can be calculated by examining the delay of a single seeded player, D_{Player_i}. The following relations are valid:

$$D_{\text{Player}_i} \in \{0, 1, \ldots, P-1\} \tag{3.20}$$

$$\Pr\left[D_{\text{Player}_i} = k\right] = \begin{cases} 0 & k \notin [0, P-1] \\ \frac{1}{P} & k \in [0, P-1] \end{cases} \tag{3.21}$$

The expected value of the delay of a seeded player therefore is $\frac{P-1}{2}$, summed over all possible numbers of seeded players and multliplied by their corresponding probabilities yields:

$$D_{\text{seeded players}} = \mathsf{u}^{(n)} \cdot (0, 1, \ldots, N) \cdot \frac{P-1}{2} \tag{3.22}$$

or for short:

$$= \mathsf{u}^{(n)} \cdot \breve{\mathsf{N}}_{\mathsf{s}} \cdot \frac{P-1}{2} \tag{3.23}$$

With $\breve{\mathsf{N}}_{\mathsf{s}} = (0, 1, \ldots, N)$. Putting together (3.19) and (3.23) finally leads to the total delay:

$$D = \mathsf{u}^{(n)} \cdot \breve{\mathsf{N}}_{\mathsf{s}} \cdot \frac{P-1}{2} + \left|\mathsf{d}^{(\mathsf{N})}\right|_1 \tag{3.24}$$

3.2 Version 2: Steady

In this section we give a modified version of the protocol described in section 3.1 on page 33 and the following to speed up the achievement of regularity.

3.2.1 Procedure

Idea

With the protocol in section 3.1, we tried to get as many participants as possible to the seeded state, as this implicates an increased throughput. Although for every choice $N \leq P$ after a sufficiently large period of time, all users will be in seeded state as we will see in section 3.3.2, in the reality this time period appears to be small enough either only for small numbers of users $N < 10$ or for a sufficiently big period-length P in relation to N. (See table 3.1 on page 44. To speed up the increase in regularity the protocol is modified in a manner the seeded players retain their state and for that reason the group of unseeded participants decreases faster.

Name of the protocol

To core idea of the protocol is to never let a seeded player leave this state. The protocol is therefore named *steady* as opposed to the transient version from section 3.1 where the seeded state could be left.

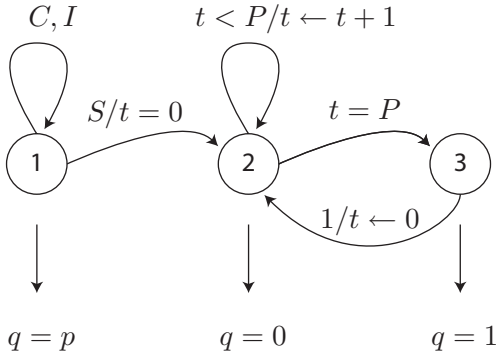


Figure 3.8. State diagram of the protocol described in section (3.2). The player stays in state 1 as long as he has not transmitted a packet successfully (which means he does not send (I) or the transmission attempt is affected by a collision (C)). As soon as a attempt succeeds (S), the user leaves the first state for ever. The player, there after, waits P-1 timeslots before sending in the Pth time slot. Even if the transmission attempt fails, he stays in the seeded stage and waits P-1 time slots once more.

Protocol description

In this method seeded players stick to their strategy after a collision and do not change to the unseeded stage. Figure 3.8 shows the state diagram of the protocol examined in this section. A player sends with probability p as long as he is unsuccessful in transmitting (S=1). As soon as a attempt succeeds, the user behaves deterministically (S=2) and sends in every Pth timeslot remaining quiet the rest of the time.

When $S = 2$ ($S = 1$) means, the considered player did (did not) transmit a packet successfully and the delay of the packet is given by i, we have:

$$q_i = \begin{cases} p & S = 1 \\ 0 & i < P,\ S = 2 \\ 1 & i \equiv 0 \mod P, S = 2 \end{cases} \tag{3.25}$$

Like in section 3.1 all N players are divided up in groups of seeded and unseeded players of size N_s and N_u respectively.

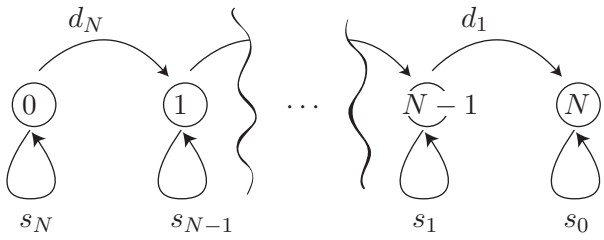


Figure 3.9. This figure shows the markov chain used in section 3.2. The states 0 to N stand for the corresponding number of seeded users. Two transition types are possible, named s and d.

3.2.2 Mathematical model

Markov chain

States The number of seeded players is again interpreted as states of a Markov chain. In contrast to section 3.1 the seeded participants never leave their state. We mark the remaining transition probabilities as in section 3.1:

$$d_{N_u} : X_{n+1} = N - N_u + 1,\ X_n = N - N_u$$
$$s_{N_u} : X_{n+1} = N - N_u,\ X_n = N - N_u$$

Transition probabilities The number of unseeded players decreases if and only if exactly one unseeded player sends, this happens with probability $N_u\, p \cdot (1-p)^{N_u - 1}$, while all seeded participants remain silent, probability $(N_u + P - N)/P$. Hence:

$$d_{N_u} = \frac{N_u + P - N}{P} \cdot N_u\, p \cdot (1-p)^{N_u - 1} \tag{3.26}$$

Since there is only one alternative transition possible we have $s_{N_u} = 1 - d_{N_u}$:

$$s_{N_u} = 1 - \frac{N_u + P - N}{P} \cdot N_u\, p \cdot (1-p)^{N_u - 1} \tag{3.27}$$

The transition matrix P is therefore given by:

$$\mathsf{P} = \begin{pmatrix} s_{N_u} & 1 - s_{N_u} & & & \\ & s_{N_u-1} & 1 - s_{N_u-1} & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 - s_1 \\ & & & & 1 \end{pmatrix} \tag{3.28}$$

At the beginning everyone starts in the unseeded state hence the starting distribution is $\mathsf{u} = (1, 0, \ldots, 0)$. After n slots the probability vector $\mathsf{u}^{(n)}$ equally $\mathsf{u} \cdot \mathsf{P}^n$.

3.2.3 Throughput

The throughput at a given number of unseeded player is denoted by $\mathrm{TP}^{\star}_{N_u}$. The same considerations as in equation (3.9) are true and we get the same equation:

$$\begin{aligned} \mathrm{TP}^{\star}_{N_u} = &\frac{(N - N_u)}{P} (1 - p)^{N_u} \\ &+ \frac{(P - N + N_u)}{P} N_u \, p \, (1 - p)^{N_u - 1} \end{aligned} \tag{3.9}$$

(See figure 3.4 on page 36. After arranging the elements of equation (3.9) as in section 3.1

$$\breve{\mathsf{TP}} = (\mathrm{TP}^{\star}_{N_u}, \mathrm{TP}^{\star}_{N_u-1}, \ldots, \mathrm{TP}^{\star}_{0}) \tag{3.10}$$

we can calculate the expected throughput at a given time instance n the same way as already stated in section 3.1:

$$\begin{aligned} \mathrm{TP} &= \mathsf{u}^{(N)} \cdot \breve{\mathsf{TP}} \\ &= \sum_{i=0}^{N} u_i^{(N)} \breve{\mathrm{TP}}_i \end{aligned} \tag{3.11}$$

The average throughput of ten thousand simulated runs in comparison with the calculated result is plotted in figure 3.10. The total number of users N as well as the period length P equal 30. The send probability of unseeded players p was set to $1/P$. Just like in figure 3.5 the simulated throughput increases in batches, but as opposed to the protocol

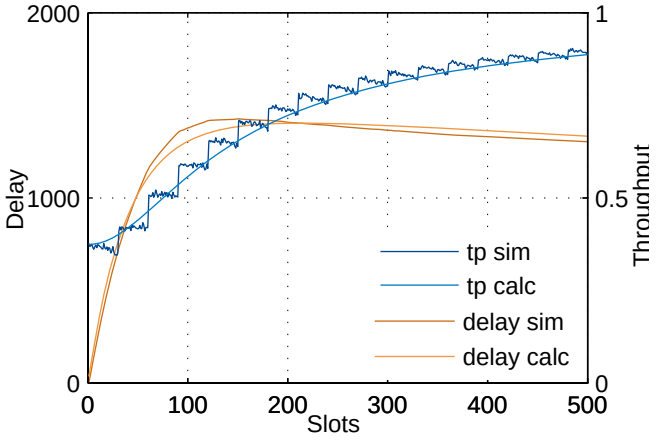


Figure 3.10. Progression of the throughput and delay for 10'000 simulated runs with period length P and participants N equal to 30. The send probability of the unseeded players p is 1/30. The calculated courses of throughput and delay are plotted as well.

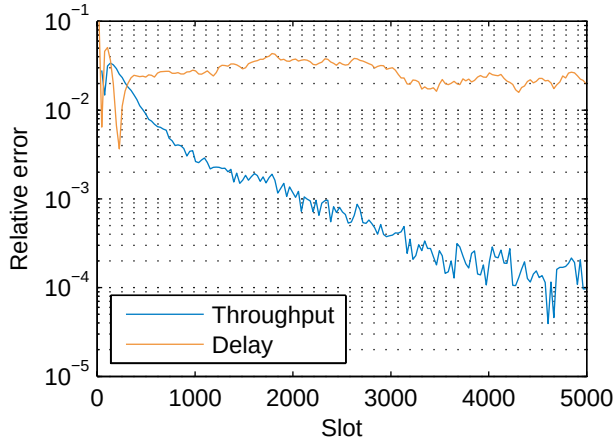


Figure 3.11. Average error of the calculated throughput and delay relative to ten thousand simulation runs with 30 players, period-length 30 and p=1/30. The error values within each period were combined (by taking the average) to compensate the periodical behavior of the throughput.

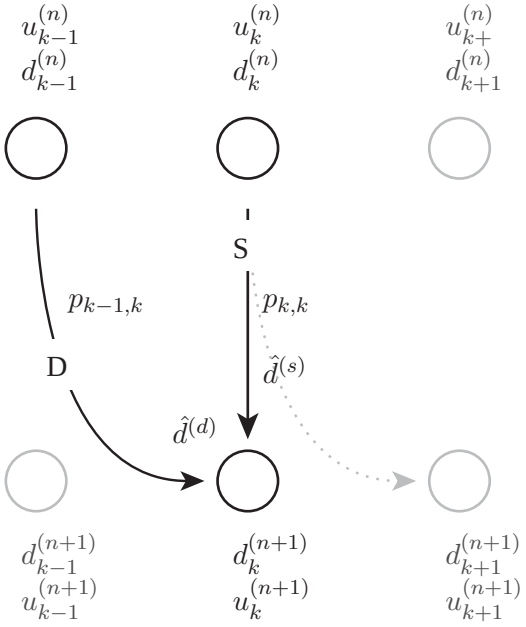


Figure 3.12. Calculation of the upcoming delay values (see section 3.2.4). $\mathsf{u}^{(n)}$ stands for the state distribution vector, d for the delay, p for the transition probabilities contained in the matrix P. The two possible transitions are called D and S.

in section 3.1 on page 33 however not sawtooth-shaped but rather cascaded.

The relative errors made by the approximation is shown in figure 3.11. The error is small from the beginning and further decreases very fast with time.

3.2.4 Delay

Figure 3.12 shows how the delay of the unseeded players is calculated. u stands for the respective state probability, d for the respective delay, p for the transition probabilities and n for the time slot. As opposed to the 3.7 the transition $k+1$ to k is no longer possible, that is why the new delay consists only of two components $\hat{d}^{(d)}$ and $\hat{d}^{(s)}$.

For the simple reason that a seeded player stay does not leave this state, the unseeded participants have to be in their state forever and be delayed since the beginning of the protocol for that reason. With these considerations follows:

$$d_k^{(n)} = u_k^{(n)} \cdot (N-k) \cdot n \tag{3.29}$$

Let $\star$ be the elementwise vector multiplication and $\breve{\mathsf{N}}_\mathsf{u}$ the vector $(N_u, N_u - 1, \ldots, 0)$, with this definition we have:

$$\mathsf{d}^{(n)} = \mathsf{u}^{(n)} \star \breve{\mathsf{N}}_\mathsf{u} \cdot n \tag{3.30}$$

The total unseeded delay is given by equation (3.19)

$$D_{\text{unseeded players}} = \left|\mathsf{d}^{(\mathsf{n})}\right|_1 \tag{3.19}$$

$$= \sum_{i=0}^{N} d_i^{(N)} \tag{3.31}$$

which can be reduced to

$$= \mathsf{u}^{(n)} \cdot \breve{\mathsf{N}}_\mathsf{u} \cdot n \tag{3.32}$$

The conditions (3.20) und (3.21) do not apply to the seeded players any more, since delays greater than P are possible. Here, nevertheless, the formula (3.22) is used as an approximation:

$$D_{\text{seeded}} \approx \mathsf{u}^{(n)} \cdot (0, 1, \ldots, N) \cdot \frac{P-1}{2}$$

$$\approx \mathsf{u}^{(n)} \cdot \breve{\mathsf{N}}_\mathsf{s} \cdot \frac{P-1}{2} \tag{3.33}$$

Adding up (3.33) and (3.19) yields:

$$D = \mathsf{u}^{(n)} \cdot \left(\breve{\mathsf{N}}_\mathsf{u} \cdot n + \breve{\mathsf{N}}_\mathsf{s} \cdot \frac{P-1}{2}\right) \tag{3.34}$$

Where $\breve{\mathsf{N}}_\mathsf{u} = (N, N-1, \ldots, 0)$ and $\breve{\mathsf{N}}_\mathsf{s} = (0, 1, \ldots, N)$.

Figure 3.10 shows how the delay calculated after (3.34) behaves in comparison with an example simulation with 30 players, a periodlength of 30 as well and $p = 1/30$. The delay is not periodical, as the throughput implies, but rather proceeds like the calculated course very smoothly.

The deviation of the calculation (3.34) from ten thousand simulation runs is drawn in figure 3.11. The biggest error appears in the example after around 100 time slots and is about 5 per cent. The relative error does, opposed to the throughput calculation, not decrease and always stays roughly between 1 and 4 percent.

3.3 Evaluation

In this section we examine the throughput and delay of the transient protocol (described in section 3.1 on page 33) referred to as version 1 as well as the steady protocol (described in section 3.2 on page 39) referred to as version 2. Both versions are finally compared with the classic ALOHA-protocol in section 3.3.4 on page 51

Furthermore it is examined how the two protocols behave after long time.

3.3.1 Long term behavior

Case $P \geq N$

Absorbing Markov chains Firstly the Markov chain of both protocols (see figures 3.2 and 3.9 on pages 35 and 40 respectively) have an absorbing state: it is impossible to leave the state N in both versions.

Secondly it is possible to get from every state to this absorbing state, since the transition probabilities d_{N_u} defined in (3.2) on page 35 and (3.26) on page 40 respectively are none zero for every choice N_u.

Therefore the Markov chains (3.5) (page 35 and (3.5) (page 41) are *absorbing Markov chains*.

To analyze absorbing Markov chains the transition matrix has to be in the so called *canonical form* [GS03]:

$$\mathsf{P} = \left(\begin{array}{c:c} \mathsf{Q} & \mathsf{r} \\ \hdashline 0 & 1 \end{array}\right) \tag{3.35}$$

$\mathsf{0}$ is a zero row vector of length N, r is a column vector of length N given by:

$$\mathsf{r} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \tag{3.36}$$

and Q a N-by-N matrix with $q_{ij} = p_{ij}$. The states 0 to $N-1$ are called transient, the last state N is absorbing.

Both the transition matrix (3.5) (on page 35) and (3.28) (on page 41) are already in this form.

Probability of absorption A theorem of the absorbing Markov chain theory says, the probability that the process will be absorbed is equal to 1 [GS03]. Since the Markov chains of both transient and steady protocol have only one absorbing state (namely the state N) we know for sure, that after a certain time the processes will be in this state N. The question is how long it lasts on an average until all users are seeded and we are therefore in state N.

Time to absorption To calculate the expected number of steps before the chain is absorbed, we set the *fundamental matrix* for P up. This matrix denoted by N is given by the following equation [GS03, Mey00]:

$$\mathsf{N} = (\mathsf{I} - \mathsf{Q})^{-1} \tag{3.37}$$

where I is a N-by-N identity matrix and Q is defined by (3.35).

Considering the steady protocol, we obtain using (3.28) and making use of the fact that $1 - s_{N_u} = d_{N_u}$:

$$(\mathsf{I} - \mathsf{Q}) = \begin{pmatrix} d_{N_u} & -d_{N_u} & & \\ & \ddots & \ddots & \\ & & \ddots & -d_2 \\ & & & d_1 \end{pmatrix} \tag{3.38}$$

The matrix (3.38) can simply be inverted by using the fact [Mey00]:

$$\mathsf{A}^{-1} = \frac{1}{\det(\mathsf{A})} \operatorname{adj}(\mathsf{A}) \tag{3.39}$$

where $\operatorname{adj}(\mathsf{A})$ is the matrix formed by the cofactors of A called the adjugate of A. Using (3.39) and (3.38) we get:

$$(\mathsf{I} - \mathsf{Q})^{-1} = \begin{pmatrix} \frac{1}{d_{N_u}} & \cdots\cdots\cdots & \frac{1}{d_1} \\ & \frac{1}{d_{N_u-1}} \quad \cdots & \frac{1}{d_1} \\ & \ddots & \vdots \\ & & \frac{1}{d_1} \end{pmatrix} \tag{3.40}$$

The fundamental matrix for the second protocol has no such neat form when expressed in terms

of the transition probabilities d (3.2), i (3.3) and s (3.4). We therefore abstain from quoting the matrix and refer to calculating the fundamental matrix numerically.

The expected number of steps before the chain is absorbed is now given by the elements t_i of the vector t:

$$\mathsf{t} = \mathsf{N}\mathsf{c} \tag{3.41}$$

where N is defined in (3.37) and c is a column vector whose entries are all 1. The element t_i of the vector t corresponds to to expected number of steps it takes to get from state s_i to the absorbing state. Since we always start in the first state, we are only interested in the value t_1. We denote this value by L:

$$L = (1, 0, \ldots, 0) \cdot \mathsf{N}\mathsf{c} \tag{3.42}$$

L is equal to the sum of the first row of N which can be explicitly written for the second (steady) protocol version:

$$L = \frac{1}{d_{N_u}} + \frac{1}{d_{N_u-1}} + \cdots + \frac{1}{d_1} \tag{3.43}$$

The first addend stands for the average time spent at the first state, the second addend for the time at the second state and so on. So on average after L time slots the protocol is in state N.

Table 3.1 shows the value of L for some N-to-P-combinations for the protocol version 1. As one sees, the L gets very big very fast if the smallest possible period length $P = N$ was chosen. In these cases, the maximum possible regularity (all player being seeded) is reached only for small groups of users < 10 foreseeable time. To get to the absorbing state for bigger groups as well, we have to increase the value of P. Table 3.1 lists the time it takes to get to the state N when $P \approx 1.5 \cdot N$ (in section 3.4 on page 53 we go further into this choice of P), as one can see the value of L doesn't increase as nearly as fast as in the first choice of P.

While in the first version of the protocol seeded players go to the unseeded state if one of their packet suffers from a collision, it's very hard to get all players seeded when N is big and P nearly equal

N	P	L	P	L
2	2	8	3	8
5	5	143	8	51
10	10	$5.8 \cdot 10^3$	15	227
15	15	$0.25 \cdot 10^6$	23	519
20	20	$11 \cdot 10^6$	30	$1.2 \cdot 10^3$
50	50	$0.24 \cdot 10^{18}$	75	$31 \cdot 10^3$

Table 3.1. Lists the time L in slots it takes to get to the absorbing state of the protocol version 1. At this state the throughput given in (3.47) is reached. N denotes the number of users, P the period length, p is set to $1/P$.

N	P	L	P	L
2	2	6	3	7
5	5	43	8	34
10	10	173	15	97
20	20	689	30	257
50	50	4234	75	852
100	100	16763	150	2020

Table 3.2. Lists the time L in slots it takes to get to the absorbing state of the protocol version 2. At this state the throughput given in (3.47) is reached. N denotes the number of users, P the period length, p is set to $1/P$.

to N (see table 3.1. In the second version of the protocol, seeded players don't leave this state. For this reason, the absorbing state is more easely reached as table 3.2 illustrates.

Case $P < N$

Second protocol version The Markov chain from section 3.2 remains absorbing. In the fundamental matrix (3.40) the main diagonal is equal to $d_{N_u}, d_{N_u-1}, \ldots, d_{N-P+1}$ and the off-diagonal $-d_{N_u}, \ldots, -d_{N-P+2}$.

The average time till the maximum throughput is reached changes to

$$L = \frac{1}{d_{N_u}} + \frac{1}{d_{N_u-1}} + \cdots + \frac{1}{d_{N-P+1}} \tag{3.44}$$

First protocol version - ergodic Markov chain If the period-length P is smaller than the number of users N, there is no absorbing state in the Markov chain for the first (transient) protocol depicted in figure 3.3 on page 35. Furthermore it's possible to go from every state to every state. The matrix P therefore describes a *ergodic* or *irreducible* Markov chain. Since P^P has no zero element the chain is also *regular* [GS03].

For these transition matrices the *fundamental limit theorem for regular chains* states [GS03]:

$$\lim_{n\to\infty} \mathsf{P}^n = \mathsf{W} \tag{3.45}$$

Where the limiting matrix W is unique and all rows are the same vector $\boldsymbol{\pi}$. So irrespective of the starting distribution the likelihood of being in of the states s_i after a infinite time is equal to π_i.

The vector $\boldsymbol{\pi}$ is called the *stationary distribution* since

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathsf{P} \tag{3.46}$$

As (3.46) shows, we can think of $\boldsymbol{\pi}$ as a left eigenvector of the transition matrix P.

3.3.2 Throughput

Case $P \geq N$

Course Figure 3.14 shows the throughput of the first and second protocol subject to the time slot, when $P = 20$, $N = 17$ and $p = 0.05$. As you can see, both protocols reach a very high throughput (given in the upcoming paragraph) the steady protocol being much faster.

Long term behaviour In section 3.3.2 we saw, that in both the transient as well as the steady protocol, the absorbing state N is reached after a known expected number of time slots. We are know interested in the throughput at this final state as well as the intermediate values when not all users are seeded yet.

The throughput according to (3.9) on page 36 then is equal to:

$$\mathrm{TP}^{\star}_N = \frac{N}{P} \tag{3.47}$$

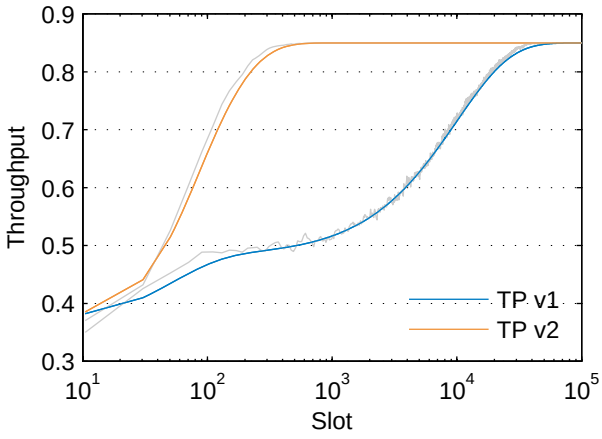


Figure 3.14. Shows the mean throughput of 10'000 simulation runs (as grey lines) for the first version as well as the second version of the protocol with $N = 17$ and $P = 20$. In blue and orange respectively the calculated results are shown. As one can see, the Markov models worked out in detail in sections 3.1 and 3.2 are well suited.

As one can easily see, when $N = P$ the throughput is maximized and reaches 1.

Since the optimal throughput you get with the classical ALOHA-protocol is approximately e^{-1} the modified ALOHA-protocol outperforms this version when (using (3.47))

$$\frac{P}{e} < N \leq P \tag{3.48}$$

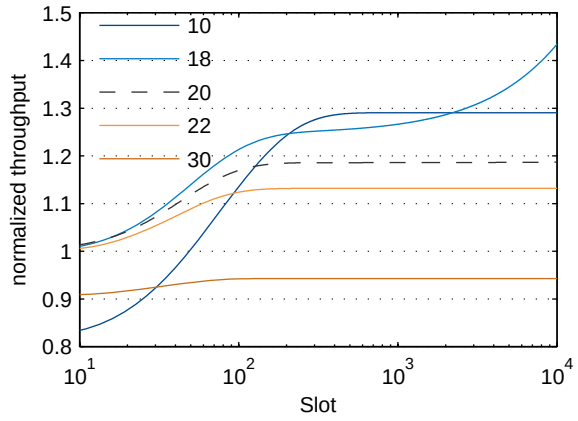
we will give a looser upper bound in equation (3.52) on page 47 when considering the case $\boldsymbol{P \geq N}$.

Case $P < N$

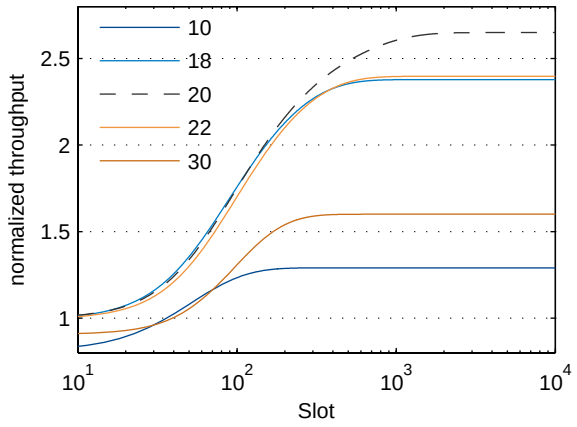
First protocol The Markov chain for the case $P < N$ is regular and does not have a absorbing state (see section 3.3.1). So the expected throughput limit is given by (3.46) and (3.11):

$$\lim_{n\to\infty} TP^{(n)} = \boldsymbol{\pi}\check{\mathsf{T}}\mathsf{P} \tag{3.49}$$

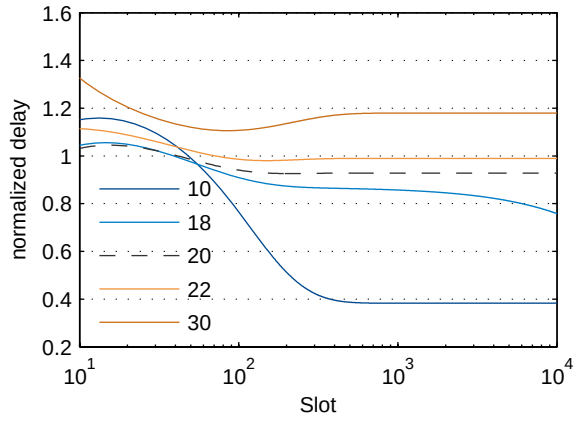
; The blue line in figure 3.15 shows the course of throughput when 23 participants use the first protocol with a period-length $P = 20$. As you can see



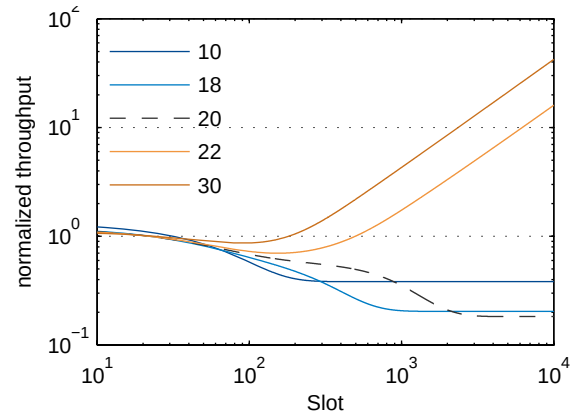
(a) Throughput of the stateless protocol (version 1): When two many users (22 or 30 in this case) are competing, the steady state π is reached after at most 100 time slots. In both cases the throughput is smaller than for the case $N = P = 20$ (intermittent line), in the case $N = 30$ the throughput is even smaller than for the conventional ALOHA-protocol.
When 10 participants compete, the absorbing state is easily reached and the throughput is equal to 0.5 according to 3.47 which beats the conventional ALOHA-protocol by about 30 percent. 18 users are to much to reach the absorbing state after 10^4 time slots, but nevertheless the normalized throughput outperforms the case $N = 10$.



(b) Throughput of the stateless protocol (version 2): Independent of the number of users the absorbing state is reached and the final throughput is equal to (3.50). In contrast to figure (a), too many users don't worsen the throughput that much.



(c) Delay of the stateless protocol (version 1): With 10,18 or 22 participants the delay is nearly equal to the conventional ALOHA-protocol, though the delay of 18 users slightly decreases with time. When only 10 users share a period of $P = 20$, they can easily reach the absorbing state and their delay is, according to 3.56a about 40 percent of the delay in the classical ALOHA protocol.



(d) Delay of the stateless protocol (version 2): In contrast to figure (c), the delay increases without limit when to many users compete. This is due to the 2 (8) players who are not seeded and therefore never succeed in sending a packet. (Please note the logarithmic scale of the y-axis).

Figure 3.13. Shows the course of the throughput (figures a and b) and delay (figures c and d) when using the first protocol and the second protocol respectively. The period length was set to 20 and the sending probability in the unseeded state p to 0.05. Normalized means, the values are divided by the respective values of the case, all users behaving according to the classical ALOHA-protocol and sending with $q = 1/N$.

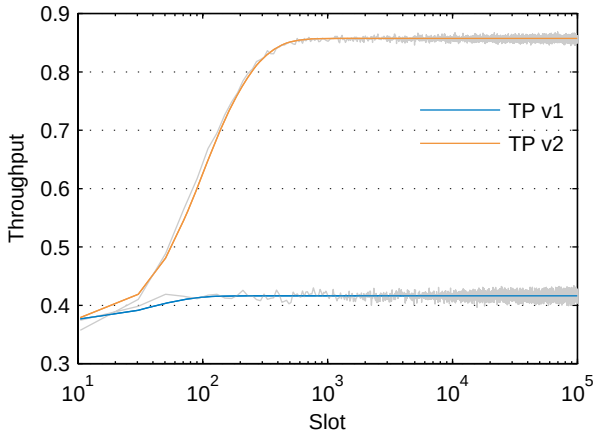


Figure 3.15. Plot of the throughput of the first and second protocol when 23 users share a period of 20 time slots. The gray lines represent the average of 10'000 simulation results. As you can see, the second protocol sends more than twice as many packets successfully than the first protocol version.

$\frac{N}{P}$	TP v1	TP v2
1.05	0.421	0.951
1.1	0.411	0.895
1.2	0.397	0.818
1.5	0.348	0.605
2	0.272	0.366
3	0.148	0.134
5	0.033	0.018

Table 3.3. Final throughput when $P < N$ holds for the transient (TP v1) and the steady protocol (TP v2). If the number of users only slightly exceed the period-length P, the throughput of the first protocol drops heavily unlike the second protocol. The sending probability p was set to $1/P$.

the values are only slightly bigger than for the conventional ALOHA-protocol (throughput ≈ 0.37).

Second protocol As already mentioned in section 3.3.1 the Markov chain of the second protocol still is absorbing. Since more users are competing than the period-length is in length, at this state P users are seeded while the remaining $N-P$ are unseeded and therefore send with probability p. In every time instance exactly one seeded player send. His packet his transferred successfully, if none of the unseeded users sends as well. Considering this the throughput is:

$$\mathrm{TP}_{N,P,p} = (1-p)^{N-P} \quad \text{where } N > P \qquad (3.50)$$

The red line in figure 3.15 shows the course of the throughput when 23 users compete using a period-length $P = 20$. As you can see, the steady protocol clearly outperforms the transient version.

Comparing (3.50) with the approximate throughput e^{-1} of the conventional slotted ALOHA–protocol we get the following condition

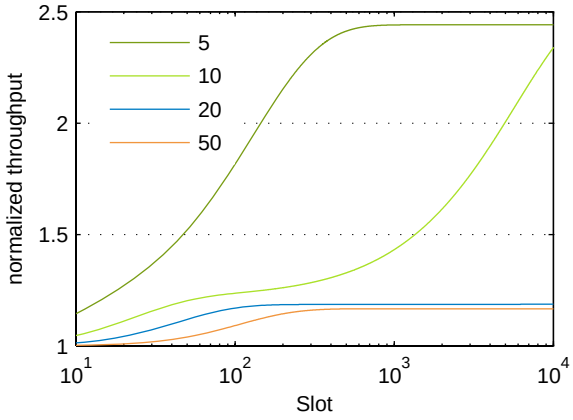
$$N < P - \frac{1}{\ln(1-p)} \qquad (3.51)$$

that must be fulfilled when the second protocol wants to beat the conventional ALOHA in terms of throughput. Using (3.51) we can give a looser upper bound for (3.48):

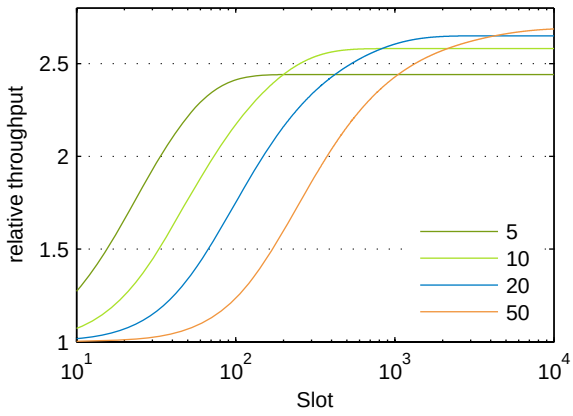
$$\frac{P}{e} < N < P - \frac{1}{\ln(1-p)} \qquad (3.52)$$

Table 3.3 shows the decrease in throughput when more users take part in the protocol than the period is in length. The first column indicates the ratio of the number of users N to the period-length P, the second column lists the achieved throughput when using the first protocol, the third column lists the throughput when using the second protocol. N was set to 100 ($p = 1/P$), but similar values result for other numbers of users.

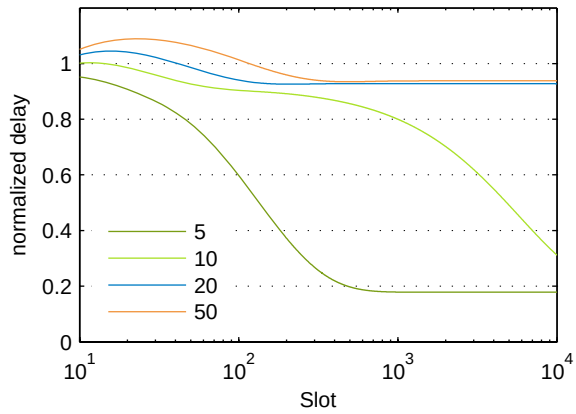
As one can see, the protocol version 1 is very sensitive to small transgression of the maximum number of users. While the second protocol only loses 1 percent of the maximum throughput when using 101 users with a period-length of 100, the first version loses 57 per cent in the same setting.



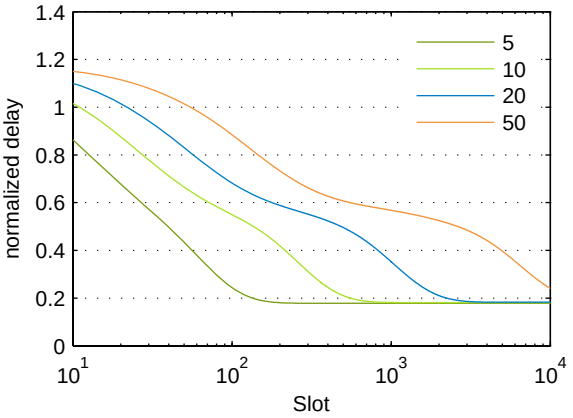
(a) Throughput of the transient protocol (version 1): As listed in table 3.1 it takes 143 (5800) slots on average until all users are seeded when there are 5 (10) participants. At this state, the throughput is equal to 1 and about 2.5 higher than for the classical ALOHA-protocol. When 20 or even 50 users compete, the seeded state cannot be reached in reasonable time. The throughput, however, is still about 20 percent higher than for the classical ALOHA-protocol.



(b) Throughput of the steady protocol (version 2): The absorbing state where all users are seeded is much faster reached (see table 3.2). Even when 50 users compete, they reach the absorbing state after approximately 4200 on average. The maximum throughput is 1, but since the throughput of the classical ALOHA-protocol is larger for smaller numbers of participants the normalized throughput is a little bit higher for 50 users than for 5 participants.



(c) Delay of the transient protocol (version 1): As already mentioned in the caption of figure (a), the absorbing state is reached after 10^4 when the number of participants is equal to 5 and 10. At this state, the mean delay is equal to (3.53) which is approximately 0.18 (see 3.56b). For greater number of users the delay is nearly of the same size as for the classical ALOHA-protocol.



(d) Delay of the steady protocol (version 2): For all numbers of participants the absorbing state is reached, where the normalized delay is equal to approximately 0.18 according to equation (3.56b)

Figure 3.16. Course of the delay and throughput of the two versions of variable-q-ALOHA protocol relative to the classic ALOHA-protocol when using the stateless protocol and the two state protocol respectively. Four different number of users are considered (N = 5,10,20,50). The periodlength equals the number of players $P = N$, p is equal to $1/N$.

Figures 3.13(a) and 3.13(b) on page 46 shows the course of the throughput for the first and the second protocol respectively.

3.3.3 Delay

In this section the total delay in the protocol versions 1 and 2 are calculated and compared with the conventional ALOHA–protocol.

Case $P \geq N$

Both the transient as well as the steady protocol have the same absorbing state and for this reason the same limiting delay.

Delay of protocol version 1 and 2 As seen in section 3.3.2 the transition matrix describes an absorbing Markov chain as long as the number of users N is less or equal the period length P. Since $N_u = 0$ and $N_s = N$ at the absorbing state, the delay only consists of the total delay of seeded players:

$$D_m = D_{\text{seeded players}}$$

which is according to (3.22)

$$= \lim_{n\to\infty} \mathsf{u}^{(n)} \cdot (0, 1, \ldots, N) \cdot \frac{P-1}{2}$$

since $\lim_{n\to\infty} \mathsf{u}^{(n)} = (0, \ldots, 0, 1)$

$$= N\frac{P-1}{2} \tag{3.53}$$

Figure 3.17 shows the course of delay of both the transient and the steady protocol. Both delays first increase and later drop to the long term value calculated above. Using first protocol, however, it takes much more slots (about a hundred times more) to reach this state.

Delay of the conventional ALOHA–protocol The expected value of the delay of a single player in the classic ALOHA-protocol after T Slots is equal

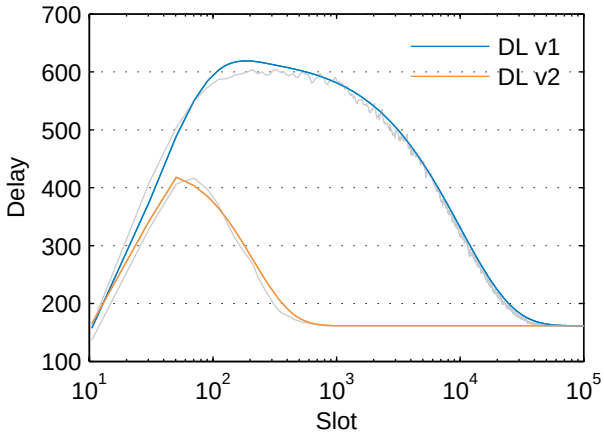


Figure 3.17. Shows the total delay for the same simulation runs used in figure 3.14. The calculated courses (blue and orange respectively) deviate a little bit (at most 3 percent) from the simulation results when the peak values are reached, but are nevertheless sufficiently precises.

to:

$$\begin{aligned} d &= \sum_{k=0}^{T-1} k \cdot (1-P_{suc})^k \cdot P_{suc} + T(1-P_{suc})^T \\ &= P_{suc}(1-P_{suc}) \sum_{k=0}^{T-1} k \cdot (1-P_{suc})^{k-1} \\ &\quad + T(1-P_{suc})^T \\ &= (1-P_{suc}) \frac{1-(1-P_{suc})^{T-1}(T - P_{suc}T - P_{suc})}{P_{suc}} \\ &\quad + T(1-P_{suc})^T \end{aligned}$$

Where P_{suc} is the probability that a particular user transmits a packet successfully. This value is given by:

$$P_{suc} = q \cdot (1-q)^{N-1}$$

where q is the send-probability (see also 1.3.2) on page 19).

The total delay D is obtained by multiplying d by N: $D = Nd$. The total delay after an infinitly

long time period equals:

$$D_c = N\frac{1-p_S}{p_S} = N\frac{1-q\cdot(1-q)^{N-1}}{q\cdot(1-q)^{N-1}} \tag{3.54}$$

This value is minimized when $q = \frac{1}{N}$ what at the same time maximizes the total throughput. For sufficiently large N when $q = \frac{1}{N}$ we have:

$$(1-q)^{N-1} \approx e^{-1}$$

And therefore

$$D_c \approx N(Ne-1) \tag{3.55}$$

Comparison When comparing the expected delay of the variable-q-ALOHA D_m with the classical ALOHA-protocol D_c when using the optimal q in the latter case we get the following result (Take into account that for the modified protocol the sending probability p is not used any more and is therefore no longer important):

$$\frac{D_m}{D_c} = \frac{P-1}{2(Ne-1)} \tag{3.56a}$$

$$\approx \frac{1}{2e} \qquad P = N \gg 1 \tag{3.56b}$$

Therefore, the modified protocol in the final state beats the classical protocol in terms of accumulated delay, when

$$\frac{P+1}{2e} < N \leq P$$

Case $P < N$

When the number of users N exceeds the period length P, the Markov chain of the first protocol is regular, while the chain of the second protocol being absorbing. For this reason we expect the delay to be different considering long term behavior.

First protocol After infinite many time slots the state probability vector is given by $\boldsymbol{\pi}$ (see (3.46)). For $n \to \infty$ the fundamental limit theorem states:

$$\lim_{n\to\infty} \mathsf{d}^{(n+1)} = \mathsf{d}^{(n)}$$

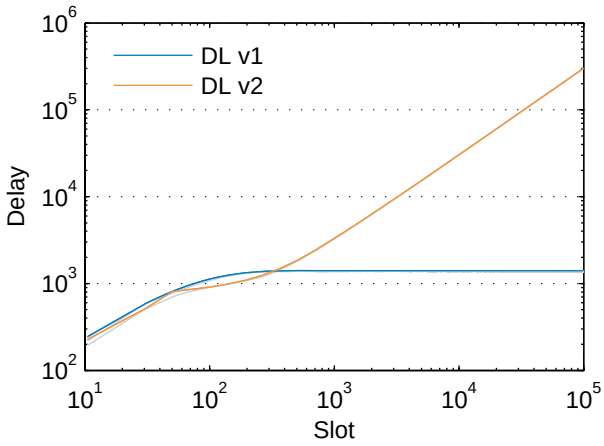


Figure 3.18. Shows the delay of the first and the second protocol when 23 N users compete and the period-length P is set to 20 time slots. The average of 10'000 simulation run is plotted as a green line but, since the calculation perfectly matches the reality, is covered by the orange and blue line. As mentioned, the delay increases unlimited when $P < N$ holds in the second protocol version.

together with equation (3.18) on page 39 the steady delay of the unseeded players δ we obtain:

$$\delta = \boldsymbol{\pi}\cdot(\mathsf{D}_+ \star \mathsf{P})\cdot(\mathsf{I}-\mathsf{P}+\mathsf{D}_- \star \mathsf{P})^{-1} \tag{3.57}$$

Using the equation (3.24) on page 39 the expected total delay for the first protocol after a infinite number of time slots is given by:

$$\lim_{n\to\infty} D^{(n)} = \boldsymbol{\pi}\cdot\check{\mathsf{N}}_{\mathsf{s}}\cdot\frac{P-1}{2} + |\delta|_1 \tag{3.58}$$

where $\boldsymbol{\pi}$ and δ are defined by (3.46) and (3.57) respectively.

The blue lines in figure 3.18 show the course of the delay when 23 participants use the first protocol with $P = 20$.

Second protocol When using the steady protocol we have P seeded players who regularly send packets and $N - P$ participants who *never* succeeded in transmitting their packets. Their delay therefore

constantly increases with time! At the absorbing the we get

$$\lim_{n\to\infty} D^{(n+1)} - D^{(n)} = N - P$$

In terms of delay, the case $\boldsymbol{P < N}$ must be avoided at all costs when using the steady protocol! The red lines in figure 3.18 show unbounded growth of the delay when 23 participants use the second protocol with $P = 20$.

Figures 3.13(c) and 3.13(d) on page 46 shows the course of the delay when using the first and the second protocol respectively.

3.3.4 Comparison with the conventional ALOHA

In this section we compare the the transient and the steady protocol with the conventional ALOHA protocol assuming long term behavior. We further assume, the participants don't know how many they are and estimate their number at 20.

The best choice in both protocols would be $P = N$ hence we use $P = 20$. The sending probability is set to $p = 0.05$.

The conventional ALOHA-protocol chooses the optimal sending probability according to $q = 1/N$ which is $q = 0.05$ for the mentioned estimation.

Transient protocol

Throughput Figure 3.19 shows the normalized throughput of the transient protocol. Normalized means, the respective values are divided by the throughput obtained by using the conventional ALOHA-protocol with the corresponding number of users. A normalized throughput of 1 therefore means, both protocol reach the same throughput while the transient protocol outperforms the standard ALOHA when the throughput is bigger than one and vice versa.

As you can see, the transient protocol outperforms the conventional ALOHA in any case, but the difference is biggest for $N \approx P$ and $N \leq P$.

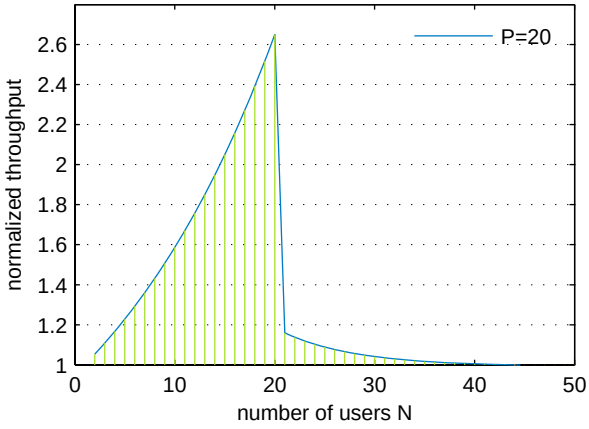


Figure 3.19. Throughput of the first version for $\boldsymbol{P = 20}$ and $\boldsymbol{p = 1/20}$ normalized by the respective throughput of the classical ALOHA-protocol where $\boldsymbol{q = 1/20}$. The ratio increases up to the maximum of $\boldsymbol{\approx 2.6}$ at $N = P$. As soon as $\boldsymbol{N > P}$ the ratio drops almost to 1.

Delay The delay values obtained were normalized the same way as the throughput, but in this case values smaller than 1 mean, the transient protocol is better than the conventional ALOHA and vice versa. Figure 3.20 shows the course of the normalized delay. As you can see, the transient protocol clearly outperforms the standard ALOHA for $N \leq P$, but results in a little bit bigger delays when $N > P$.

Steady protocol

Two compare the throughput and delay of the steady protocol, we use the same normalizing as in section 3.3.4.

Throughput Figure 3.21 shows the normalized throughput. The results for number of participants $N \leq P$ tally with the results using the transient protocol. But instead of dropping fast thereafter, the throughput decreases in a slower manner.

Delay Figure 3.22 shows the normalized delay. Again the values for $N \leq P$ tally with the result

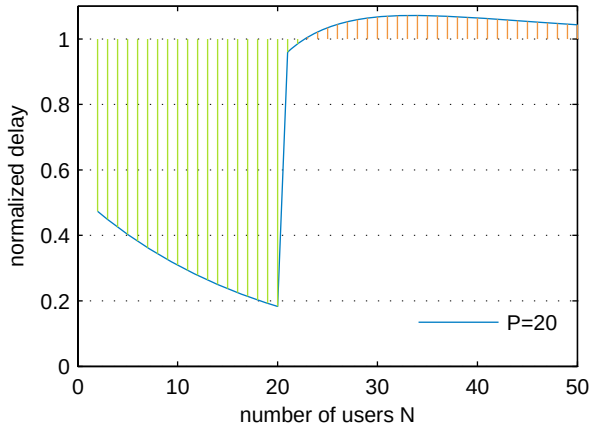


Figure 3.20. Shows the ratio of the delay in the stateless protocol (version 1) to the delay in the conventional aloha protocol for different numbers of participants. The period-length P is set to 20. The sending probability in the classical ALOHA-protocol q as well as the probability q is set to 1/20.

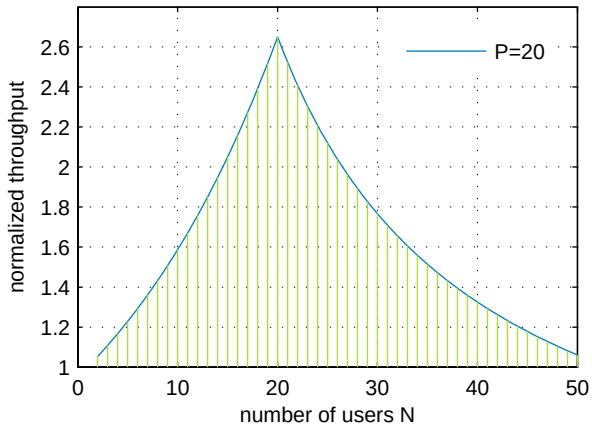


Figure 3.21. Throughput ratio for the same setting as in figure 3.19. The results for numbers of participants N smaller or equal the period-length P tallies with the result for the first version. But instead of dropping fast, the throughput ratio decreases in a slower manner.

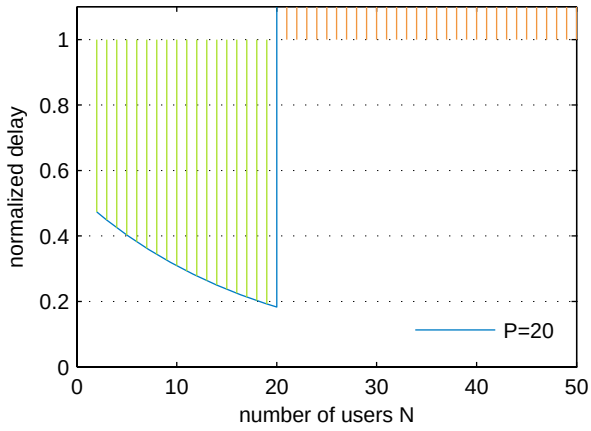


Figure 3.22. Shows the ratio of the delay in the two state protocol (version 2) to the delay in the conventional aloha protocol for different numbers of participants. The period-length P is set to 20. The sending probability in the classical ALOHA-protocol q as well as the probability q is set to 1/20.
For $1 \leq N \leq 20$ the results tallies with the results in figure 3.20, when $N > P$ the delay grows unlimitedly and the delay ratio $\longrightarrow \infty$.

using the transient protocol. When the number of users exceeds the period-length, the delay grows to infinity and so does the normalized delay.

3.4 Optimal period length P

For this reason it may be wise to choose the period length more generously, especially when using the first protocol and large numbers of participants. By increasing P, the maximum throughput decreases to the value N/P according to equation (3.47). But on the other hand this value is more easily reached, since there are more "'seeded states"' to be shared among the unseeded users. The same thing holds for the delay, the final value increases from $N(N-1)/2$ to $N(P-1)/2$ (see (3.53)), but the increase speed to absorbing state might offset this deterioration.

The question now is: Given a number of participants N and either the first or the second protocol according to 3.1 and 3.2 respectively, what is the optimal choice of P such that:

First Question: The delay is minimized.

Second Question: The throughput is maximized.

after T time slots. Clearly if $T \to \infty$ the answer to both question is $P_{opt} = N$. Figure 3.23 show the best choice of P considering the delay and the throughput of the first and second protocol respectively. The number of participants is set to 50.

Considering the delay using the first protocol, the period length should almost be doubled. When it comes to throughput, the optimum period length is about 70 percent bigger than the number of users. Both scales don't decrease much when waiting a longer time $T > 2000$.

If we look at the second protocol and try to optimize the delay, the period length must be increased by 40 percent at most when measuring after 500 time slots. The ideal additive decreases fast in the course of time and is already under 10 percent after 5000 time slots. Considering the throughput, an elevated period length is never advisable, for small measuring times $T < 500$ the optimum is reached in fact for $P < N$!

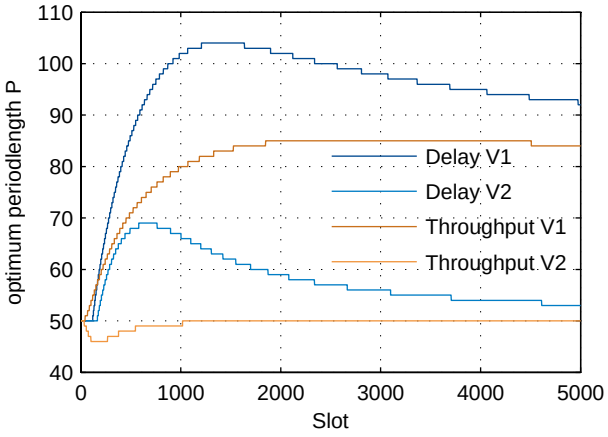


Figure 3.23. Shows the optimal period length P after a given time when 50 participants use the first and the second protocol version respectively. In blue, the optima in terms of delay are given, while the red line represent the optima considering the throughput.
As one can see, to minimize the delay bigger period-lengths P are necessary than when minimizing the throughput. Furthermore the values considering the second protocol are much lower.

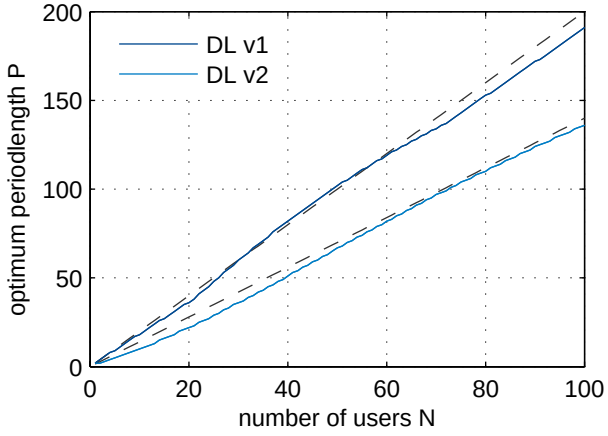


Figure 3.24. Shows the optimum value of P when considering the delay after 1000 time slots subject to the number of participants N for the first and second protocol respectively. The dotted lines represent the ratio 2 (first protocol) and 1.4 (second protocol).

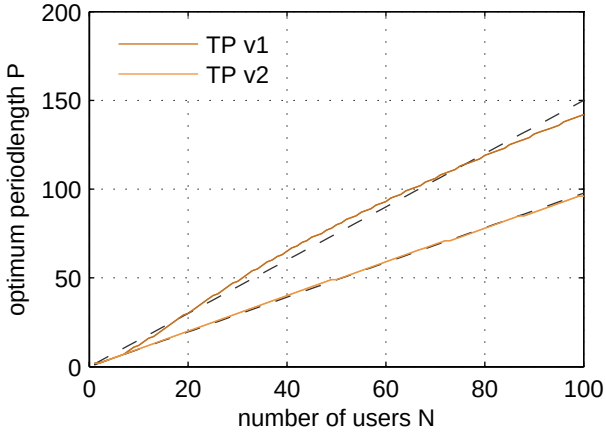


Figure 3.25. Plot of the optimum values of the period-length P in terms of throughput for the first and second protocol respectively. The dotted lines represent the ratio 1.5 (first protocol) and 1 (second protocol).

To examine the optimum period length P for various numbers of participants, we set the decisive time instance T to 1000 slots. Figure 3.24 shows the optimum values of P when considering the delay. As already mentioned, the values for the first version of the protocol are significantly larger than for the second version. The ratio P_{opt}/N is almost constant:

$$\frac{P_{\text{opt DL,V1}}}{N} \approx 2 \qquad \frac{P_{\text{opt DL,V2}}}{N} \approx 1.4$$

Figure 3.25 shows the optimum period-length in terms of throughput. As for the delay, the ratios P_{opt}/N are almost constant:

$$\frac{P_{\text{opt TP,V1}}}{N} \approx 1.5 \qquad \frac{P_{\text{opt TP,V2}}}{N} \approx 1$$

3.5 Employment of the variable-q-strategies by unfair participants

In this section, it will be examined how the strategies from section 3.1 and 3.2 on pages 33 and 39 respectively can be employed by unfair participants. The remaining fair participants behave according to the classic ALOHA-protocol described in chapter 1.

3.5.1 Employment of the transient protocol

Markov chain

N users participate altogether - N^f of them behave according to the classic Aloha protocol with send probability q and the remaining $N - N^f = N^u$ of them according to the protocol from section 3.1. These N^u participants are further divided into a group of the seeded players (size N_s) and a group of the free players (size N_u).

The number of seeded users N_s is interpreted as states of a Markov chain like in the section 3.1.2 on page 34. However in opposition to that the chain is ergodic even for the case $P \leq N^u$.

Because of the fair participants, the three transition probabilities (3.2), (3.4) and (3.3) change. In order to decrease the number of unseeded players, exactly one of them must send (probability $N_u\, p\, (1-p)^{N_u-1}$), and all participants seeded already must remain quiet (probability $1 - \frac{N_s}{N}$). Additionally, however, all the fair players mustn't send

as well (probability $(1-q)^{N^f}$. Hence equation (3.2) becomes:

$$d_{N_u} = \frac{(P - N^u + N_u)}{P} N_u\, p\, (1-p)^{N_u - 1} \cdot (1-q)^{N^f} \tag{3.59}$$

(Please note that q stands for the send probability of the fair users while p denotes the send probability of unfair users who are at the unseeded stage.) The increase in number of unseeded players occurs, when a seeded player sends ($\frac{N_s}{N^u}$) and at the same time, at least an unseeded *or* fair participant sends as well. Therefore equation (3.3) turns to:

$$i_{N_u} = \frac{(N^u - N_u)}{P} \left(1 - (1-p)^{N_u} (1-q)^{N^f}\right) \tag{3.60}$$

The probability s can be obtained simply by substracting (3.59) and (3.60) from 1.

$$s_{N_u} = \frac{(P - N^u + N_u)}{P} \cdot \left(1 - N_u\, p\, (1-p)^{N_u - 1} (1-q)^{N^f}\right) + \frac{(N^u - N_u)}{P} (1-p)^{N_u} (1-q)^{N^f} \tag{3.61}$$

Which means either no unseeded player transmits a packet successfully while no seeded player sends or a seeded player makes a successful transmission.

With the transition probabilities (3.59), (3.60) and (3.61) we can compile the transition probability matrix (3.5) (if $N \leq P$) or (3.6) if ($N > P$). (In this section $i_0 > 0$ and $s_0 < 1$ due to the existence of fair players).

Throughput

For the total average throughput of unfair participants, we need the probability distribution and, on the other hand, the throughput for a given number of unseeded players. At beginning, no one of the (unfair) participants is at the seeded stage. The starting distribution therefore is $\mathsf{u} = (1, 0, \ldots, 0)$ as in section 3.1. After n time slots we have $\mathsf{u}^{(n)} = \mathsf{u} \cdot \mathsf{P}^n$ (*see equation* (3.8)). The throughput of unfair players for a certain number of unseeded players corresponds to the likelihood that a seeded player (first summand) or a unseeded player (second summand) make a successful transmission attempt:

$$\mathrm{TP}^{\star\text{unfair}}_{N_u} = \frac{(N^u - N_u)}{P} (1-p)^{N_u} (1-q)^{N^f} + \frac{(P - N^u + N_u)}{P} N_u\, p\, (1-p)^{N_u - 1} \cdot (1-q)^{N^f} \tag{3.62}$$

Note that equation (3.62) can be obtained by multiplying (3.9) by $(1-q)^{N^f}$. After putting together the elements (3.62) in descending order in the vector $\breve{\mathsf{TP}}^{\text{unfair}} = (\mathrm{TP}^{\star\text{unfair}}_{N_u}, \mathrm{TP}^{\star\text{unfair}}_{N_u-1}, \ldots, \mathrm{TP}^{\star\text{unfair}}_{\max(N-P,0)})$ the total throughput of unfair players after n time slots is $\mathrm{TP}^{\text{unfair}} = \mathsf{u}^{(n)} \cdot \breve{\mathsf{TP}}$.

A fair player transmits a packet successfully, if and only if:

1. He is the only fair player who sends (probability $q(1-q)^{N^f-1}$).
2. No seeded player sends (probability $1 - N_s/P$).
3. No unseeded player sends (probability $(1-p)^{N_u}$).

After replacing N_s by $N^u - N_u$, multiplying the three probabilities above we get the total throughput of fair players for a given number of unseeded (unfair) participants after multiplying the result by N^f:

$$\mathrm{TP}^{\star\text{fair}}_{N_u} = N^f q (1-q)^{N^f-1} \cdot \left(1 - \frac{N^u - N_u}{P}\right) \cdot (1-p)^{N_u} \tag{3.63}$$

The total throughput of the group of fair participants can be obtained by multiplying the probability distribution $\mathsf{u}^{(n)}$ by the vector $\breve{\mathsf{TP}}^{\text{fair}} = (\mathrm{TP}^{\star\text{fair}}_{N_u}, \mathrm{TP}^{\star\text{fair}}_{N_u-1}, \ldots, \mathrm{TP}^{\star\text{fair}}_{\max(N-P,0)})$.

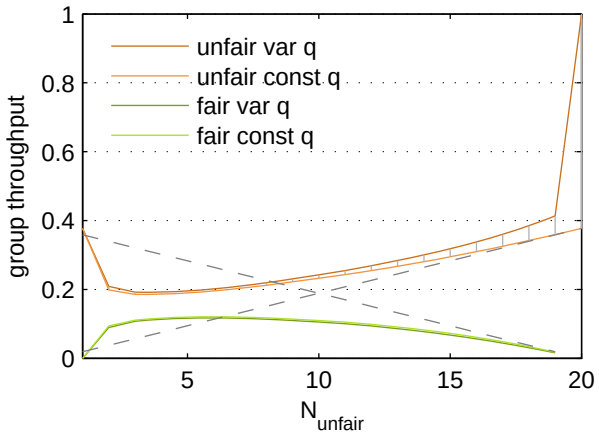


Figure 3.26. Shows the total throughput of unfair and fair players using the variable-q-protocol from section (3.5.1) as well as the optimal constant-q-protocol from section (1.4.1) on page 19. The dotted lines shows the throughput that would result, when all players beeing fair and sending with constant probability $q = 1/N$. The total number of users was 20, the fair users did send with probability 1/20. The unfair players used $P = N^u$ and $p = 1/P$.

Figure 3.26 shows the total throughput of unfair players, when they behave like it is described in this section. The period-length was set to the number of unfair users N^u and $p = \frac{1}{P}$. The fair players used a q equal to $1/N$. In lighter colors, the throughput of the protocol form section 1.4.1 on page 19 is shown. As you can see, the variable-q-protocol does worsen the throughput of fair players nearly the same way as the protocol from section 1.4.1 does but enhances the throughput of unfair players a little bit more, especially for higher fractions (shown with light gray bars).

Delay

Given the transition probabilities (3.59), (3.60) and (3.61) we can calculate the delay of unseeded players according to section 3.1.4 on page (36).

3.5.2 Employment of the steady protocol

The steady protocol from section 3.2 on page 39 can also be employed by unfair users. As we did in section 3.5.1, we first have to modify the transition probabilities of the Markov chain. The throughput and delay are given thereafter.

Markov chain

Again, we set up the Markov chain at first. The participants are divided into the group of the fair and unfair participants like we did in section 3.5.1, the latter of which is divided further into seeded and unseeded players. Even with the existence of fair participant the number of seeded players cannot decrease. That's why we only have to calculate two state transitions.

A decrease of unseeded players only takes place if exactly one of them sends and both the seeded as well as the unseeded participants remain silent. Hence:

$$d_{N_u} = \frac{N_u + P - N^u}{P} \cdot N_u\, p \cdot (1-p)^{N_u - 1} \cdot (1-q)^{N^f} \tag{3.64}$$

So the only difference to equation (3.26) is the factor $(1-q)^{N^f}$. The probability to remain at a stage clearly is equal to $1 - d_{N_u}$:

$$s_{N_u} = 1 - \Big(\frac{N_u + P - N}{P} \cdot N_u\, p \cdot (1-p)^{N_u - 1} \cdot (1-q)^{N^f} \Big) \tag{3.65}$$

The transition matrix P is given in equation (3.28) on page 41 where s is given in (3.65) and d has been replaced by $1 - s$. The last state N (or P if $N > P$) is absorbing and so is the whole Markov chain.

The probability vector $\mathsf{u}^{(n)}$ again equals $\mathsf{u}\mathsf{P}^n$ where u denotes the starting distribution $(1, 0, \ldots, 0)$.

Throughput

The throughput for a given number unseeded player at an infinite time instance is denoted by $\mathrm{TP}^{\star\text{unfair}}_{N_u}$ and corresponds to the probability either a package of a seeded player (first addend in the following equation) or a package of a unseeded player (second addend) is transferred successfully and we get the same result as in (3.62) on page 55. The total throughput of unfair participants results after combining the elements (3.62) the same way we did in section 3.5.1.

The same considerations as in section 3.5.1 also apply for the total throughput of fair participants at a particular N_u and hence the throughput is obtained by combining the elements (3.63) on page 55 and multiplying the result by the probability distribution $\mathrm{u}^{(n)}$ afterwards.

Figure 3.27 shows the throughput of the unfair players, that results from beeing in the absorbing state. In this state, all (unfair) users are seeded as long as $P \geq N$ holds. Therefore, the total throughput is given by (3.62):

$$\lim_{n\to\infty} \mathrm{TP}^{(n)}_{\text{unfair}} = \mathrm{TP}^{\star\text{unfair}}_{\max(0,N-P)}$$

what becomes $\forall P \geq N$:

$$= \frac{N^u}{P} \cdot (1-q)^{N^f} \tag{3.66}$$

and $\forall P < N$:

$$= (1-p)^{N^u-P}(1-q)^{N^f} \tag{3.67}$$

The throughput of fair users is equal to:

$$\lim_{n\to\infty} \mathrm{TP}^{(n)}_{\text{fair}} = \mathrm{TP}^{\star\text{fair}}_{\max(0,N-P)}$$

where $\mathrm{TP}^{\star\text{fair}}$ is given in (3.63). Hence the throughput becomes:

$$= N^f q(1-q)^{N^f-1}\left(1-\frac{N^u}{P}\right) \tag{3.68}$$

when $P > N$ and

$$= 0 \tag{3.69}$$

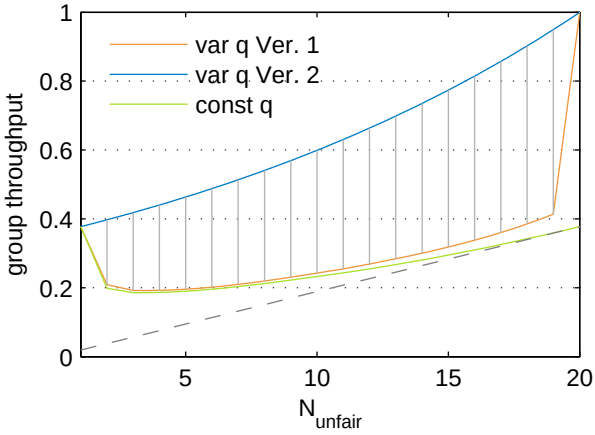


Figure 3.27. Shows the total throughput of the unfair group when they behave according to the protocol described in section (3.5.2). As comparison, the throughput was drawn, that the unfair group reaches if it behaves like indicated in sections (3.5.1) and (1.4.1) respectively. The dotted lines shows the throughput that would result, when all players beeing fair and sending with constant probability $q = 1/N$. The total number of users was 20, the fair users did send with probability 1/20. The unfair players used $P = N^u$ and $p = 1/P$.

when $P \leq N$

The throughput of all fair participants therefore decreases by the factor $(1 - \frac{N^u}{P}$ when $N < P$ and become zero as soon as $N \geq P$.

Since we know now which throughput results from the absorbing state, we want to examine (like we did in section 3.3.2 on page 45 how long it lasts on average until this state is reached.

To answer this question, we calculate the fundamental matrix N according to (3.37) on page 43. We get the same result, since Q has the same structure even with the presence of fair participants. Therefore L is given by the same equation (3.43) on page 44. As we already mentioned before the probabilities d given in (3.64) are equal to (3.26) on page 40 except for the constant factor

$$\alpha = (1-q)^{N^f} \tag{3.70}$$

On the basis of (3.43) we see, that the average time until the unfair players reach the absorbing state is increased by the factor α due to the existance of fair users.

Delay

The packets of the unseeded players are delayed since the beginning and we get the same total delay of unseeded player as in section 3.2:

$$D_{\text{unseeded players}} = \mathsf{u}^{(n)} \cdot \check{\mathsf{N}}_{\mathsf{u}} \cdot n \tag{3.32}$$

As in section 3.2 it may take multiple times to for a seeded player to transmit his packet successfully. Since N^f send with probability q at every time instance, the probability a packet of a seeded player collides is

$$P_{col}^{\text{seeded}} \approx 1 - (1-q)^{N^f} \tag{3.71}$$

where we ignored the unseeded players. Given the probability (3.71) we can calculate the expected number of attempts $\mathcal{A}$ a seeded users needs to transmit a packet:

$$\begin{aligned} \mathcal{A} &= \frac{1}{P_{col}} \\ &= \frac{1}{1-(1-q)^{N^f}} \end{aligned} \tag{3.72}$$

The total delay of seeded players can now be estimated by multiplying the result (3.33) on page 42 by $\mathcal{A}$:

$$D_{\text{seeded}} \approx \mathsf{u}^{(n)} \cdot \check{\mathsf{N}}_{\mathsf{s}} \cdot \frac{P-1}{2} \cdot \mathcal{A} \tag{3.73}$$

When $N^u \leq P$ the delay ist therefore increased by $\mathcal{A}$ given in (3.72).

Chapter 4

Cost models

4.1 Two simple introductive examples

In this section, two simple cost-models are examined: Firstly, costs per transmission attempt and secondly, costs for each time slot a packet is delayed, called delay costs.

4.1.1 Transmission costs

For each transmission attempt, costs of quantity α arise. For each successful transmission a participant is credited with one income unit. Let the inspected player 1 send with constant probability q' and the remaining $N-1$ players with probability q. Therefore costs of size $q'\alpha$ per time slot arise. The probability of a successful transmission - which is equal to the income per time slot - is equal to $q'(1-q)^{N-1}$. Therefore the gain function is:

$$E_{\text{per slot}} = q'\big((1-q)^{N-1} - \alpha\big)$$

(Please note, that *gain* is defined as the difference between income and costs: $E_{\text{per slot}}$ = income per slot - costs per slot).

Three different cases depending on q exist:

$(1-q)^{N-1} > \alpha$: The expected gain is proportionally to the sending probability of the first player. The best choice for player 1 therefore is to constantly send.

$(1-q)^{N-1} = \alpha$: The gain is always zero, independent of the choice q'.

$(1-q)^{N-1} < \alpha$: The gain $E_{\text{per slot}}$ can be at most 0 (when $q' = 0$), since the sending costs are bigger than the expected gain. Therefore player 1 will not send any packet.

4.1.2 Delay costs

In this section the following model should be examined:

- For each successfully sent packet a income of 1 is credited.
- For each time slot, a packet has to wait for it's transmission, the user is charged with costs of the scale of α.

A packet which is delayed by d time slots and then sent therefore leads to the following gain per time slot:

$$E_{\text{per slot}} = \frac{1-\alpha\cdot d}{d+1} \tag{4.1}$$

The probability that player 1 sends is constant and equal to q'. The remaining $N-1$ players send with probability q. If player 1 sends, his packet will collide (P'_{col}), if at least one of the remaining participants sends, hence $P'_{col} = 1-(1-q)^{N-1}$. On the other hand, in a given time slot player 1 transmits a packet with probability $q'(1-P'_{col})$.

Let the random variable D denote the number of time slots the packets of player 1 are delayed before transmission. The probability distribution of D is equal to:

$$\Pr[D=d] = \big[1-q'(1-P'_{col})\big]^d \cdot q'(1-P'_{col}) \tag{4.2}$$

The expected gain $E_{\text{per slot}}$ using (4.1) and (4.2) is given as:

$$\begin{aligned}\mathrm{E}[E_{\text{per slot}}] &= \sum_{d=0}^{\infty} \Pr[D=d] \cdot \frac{1-d\alpha}{d+1} \\ &= q'(1-P'_{col}) \sum_{d=0}^{\infty} \Big(\big[1-q'(1-P'_{col})\big]^d \\ &\quad \cdot \frac{(1+\alpha)-(d+1)\alpha}{d+1} \Big)\end{aligned}$$

After substituting $[1-q'(1-P'_{col})] = p$ and $q'(1-P'_{col}) = C$ we have:

$$\mathrm{E}[E_{\text{per slot}}] = C(1+\alpha) \sum_{d=0}^{\infty} p^d \frac{1}{d+1} \tag{4.3}$$

$$- C\alpha \sum_{d=0}^{\infty} p^d \tag{4.4}$$

The second added of the latter equation (4.4) stands for a geometric series. The series represented by the first added (4.3) can be simplified by using [Pap01]:

$$\ln\left(\frac{1}{1-x}\right) = \frac{x^1}{1} + \frac{x^2}{2} + \frac{x^3}{3} + \dots$$

Finally the expected gain is:

$$\mathrm{E}[E_{\text{per slot}}] = C(1+\alpha)\frac{1}{p}\ln\left(\frac{1}{1-p}\right) \tag{4.5}$$

$$- C\alpha\frac{1}{1-p} \tag{4.6}$$

with

$$\begin{aligned} C &= q'(1-P'_{col}) \\ p &= \big(1-q'(1-P'_{col})\big) \end{aligned}$$

4.2 Delay dependent sending probability

As in chapter 3 on page 33 the players send with a delay-dependent probability (the send probability depends on the delay, that the current package waits for transmission). Here, however, effects, that the players fall into some kind of deterministic state in which they send alternately (see chapter 3) by a clever choice of the sending probabilities q_i, should be avoided. Therefore, after each transmission the participant has to wait for a random amount of time, modeled by a Poisson process.

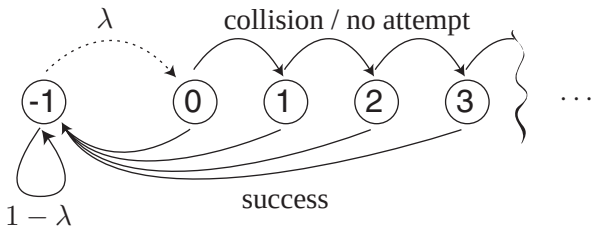


Figure 4.1. Illustrates the model used in this section. At stage -1 the participant waits for a packet to be generated, if this happens, the user tries to send the packet immediately within the same time slot (hence the dotted line). As long as the packet is not transferred successfully, the delay (giving the name to the states) is increased and the user goes to a higher state. As soon as a transmission attempt succeeded, the user returns to the waiting state -1.

4.2.1 Variable names

Each participant i ($1 \le i \le N$) either has a packet to send and his state is given by the number of time slots j passed since the arrival of the packet, $s^{(i)} = j, j \ge 0$, or he waits for a packet and is in the state $s^{(i)} = -1$. Therefore the following states are possible:

$$\mathsf{s}^{(i)} = \{-1, 0, 1, \dots\} \quad 1 \le i \le N$$

Figure 4.1 shows the model used in this section. λ defines the time a users has to wait for a new packet. When a new packet arrives, the user tries to send it in the same time slot (dotted line).

The delay of player i at a given state $s^{(i)} = j$ is equal to j if $j \ge 0$ and is not defined if $j = -1$.

$p_k^{(i)}$ denotes the probability the player i is at the state $s^{(i)} = k$. Let the sending probability of player

i at a given state be:

$$\Pr\left[\text{Player i sends}|s^{(i)}=k\right]=q_k^{(i)}$$

Clearly, $q_{-1}^{(i)}=0$. If player i tries to transmit a packet in the current time slot, we set $A^{(i)}=1$ and $A^{(i)}=0$ otherwise. $P_{col}^{\prime(i)}(k)$ stands for the probability of a packet collision, if player i at state k sends ($P_{col}^{\prime(i)}(k)=\Pr\left[\text{collision}|A^{(i)},s^{(i)}=k\right]$.

Finally, the random variable $D^{(i)}$ denotes, how long a packet of player i has to wait on average until successful transmission. $0\leq D^{(i)}$

4.2.2 Calculations

In the following we assume, every participant behaves the same way and therefore we can omit the player-index i in most cases. With $\Pr[D=d]$ denoting the probability a packet is delayed by exactly d time slots before transmission and $\mathrm{E}[W]$ standing for the expected waiting time at the state -1 we can calculate the probability, that a user is at state k:

$$p_k=\frac{\sum_{d=k}^{\infty}\Pr[D=d]}{\mathrm{E}[W]+\sum_{d=0}^{\infty}(d+1)\Pr[D=d]}\quad\forall k\geq 0$$

and with $\sum_{d=0}^{\infty}d\Pr[D=d]$ being the expected delay $\mathrm{E}[D]$:

$$=\frac{1}{\mathrm{E}[W]+\mathrm{E}[D]+1}\cdot\sum_{d=k}^{\infty}\Pr[D=d]\quad\forall k\geq 0 \tag{4.7}$$

The share of time spent waiting for new packets is:

$$p_{-1}=\frac{\mathrm{E}[W]}{\mathrm{E}[W]+\mathrm{E}[D]} \tag{4.8}$$

Since being in state k means, the player was at state $k-1$ before, the values of p_k are monotonic increasing $\forall k\geq 0$. On the other hand, p_{-1} can be smaller than any other p_k due to the fact, that the player directly goes to state 0 if a packet is generated.

Since a player at state -1 does not generate a new packet with likelihood $1-\lambda$, the expected number of time slots spent waiting is:

$$\mathrm{E}[W]=\frac{1}{\lambda}-1 \tag{4.9}$$

(Please note, since the user can send a generated packet within the same time slot, the addend -1 occurs).

To simplify the calculations we assume, that $P_{col}^{\prime}(k)$ does not depend on the state k. A packet being sent after d time slots means, the packet has been delayed for d time slots and successfully sent in the $d+1$th time slot. Therefore we have:

$$\begin{aligned}\Pr[D=d]=&\left((1-q_0)+q_0P_{col}^{\prime}\right)\\&\cdot\left((1-q_1)+q_1P_{col}^{\prime}\right)\\&\cdots\\&\cdot\left((1-q_{d-1})+q_{d-1}P_{col}^{\prime}\right)\\&\cdot\left(q_d(1-P_{col}^{\prime})\right)\end{aligned}$$

what is equal to

$$\begin{aligned}\Pr[D=d]=&\prod_{k=0}^{d-1}\left(1-q_k(1-P_{col}^{\prime})\right)\\&\cdot\left(q_d(1-P_{col}^{\prime})\right)\end{aligned} \tag{4.10}$$

After having calculated the probability distribution p_k (equations (4.7) and (4.8)) and the distribution of the delay D (4.10) there is a third value that ought to be considered, the collision probability $P_{col}^{\prime}$:

$$\begin{aligned}P_{col}^{\prime}&=\Pr\left[\text{collision}|A^{(j)}=1\right]\\&=1-\prod_{\substack{i=1\\i\neq j}}^{N}\Pr\left[A^{(i)}=0\right]\\&=1-\left(\Pr\left[A^{(i)}=0\right]\right)^{N-1}\\&=1-\left(\sum_{k=0}^{\infty}\left(p_k(1-q_k)\right)\right)^{N-1}\end{aligned} \tag{4.11}$$

Where $\sum_{k=0}^{\infty}\left(p_k(1-q_k)\right)$ can be interpreted as the expected sending probability of any player.

Therefore the dependencies of the equations (4.7), (4.10), (4.11) and the parameters q and λ are (see figure 4.2):

- $p_k = f(D,\lambda)$

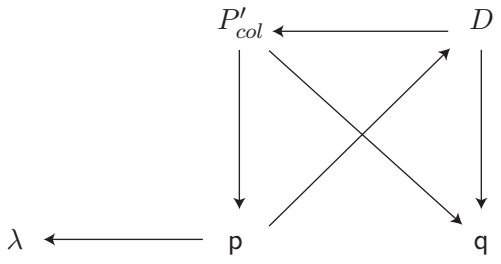


Figure 4.2. Shows the dependencies of the equations (4.7), (4.10), (4.11) and the parameters q and λ on each other. An arrow from A to B means, the equation for A contains B. As we can see, by substitution no formula for (4.7), (4.10) or (4.11) can be given, such that it only contains q and λ

- $D = g(\mathsf{q}, P'_{col})$
- $P'_{col} = h(\mathsf{p}, \mathsf{q})$

p and q stand for the vectors respectively $[p_0, p_1, \ldots]$ and $[q_0, q_1, \ldots]$. Due to the mutual dependencies of the equations derived in this section, no formula for the probability distribution of D (4.10) depending only on the parameters λ and q can be given, but distribution clearly can be calculated using numerical approaches.

4.2.3 Special relations

The inverse of the sum of the mean delay increased by 1 and the mean waiting time $\mathrm{E}[W]$ equals the throughput TP:

$$\mathrm{TP} = \frac{1}{\mathrm{E}[D] + \frac{1}{\lambda}} \tag{4.12}$$

The probability of being at state p_0 is equal to the inverse of the mean delay increased by one:

$$p_0 = \frac{1}{\mathrm{E}[D] + 1} \tag{4.13}$$

4.2.4 General cost model

Now that we know the distribution of the delay D we define the costs of packet transmissions as follows:

$$C = \sum_{k=0}^{\infty} \alpha_k \cdot p_k \tag{4.14}$$

For every successful transmission, γ units are credited. The task now is to maximize the gain per time slot:

$$G = \gamma \mathrm{TP} - \alpha_k \cdot p_k \tag{4.15}$$

We focus on two choices of α_k:

First cost model

$$\alpha_k = \begin{cases} 0 & k < \mathcal{D} \\ 1 & k \geq \mathcal{D} \end{cases} \tag{4.16}$$

Hence, for every time slot a packet is overdue regarding a deadline $\mathcal{D}$, one cost unit is charged.

Second cost model Secondly we consider the cost model that for every packet passing the deadline $\mathcal{D}$ one cost unit is charged, independently from whether the deadline is passed by only one time slot or by a huge amount of time slots. This model is defined by:

$$\alpha_k = \begin{cases} 0 & k < \mathcal{D} \\ 1 & k = \mathcal{D} \\ 0 & k > \mathcal{D} \end{cases} \tag{4.17}$$

4.3 Results

4.3.1 Analyzed sending probabilities

Three possible types of sending probability vectors q now should be examined.

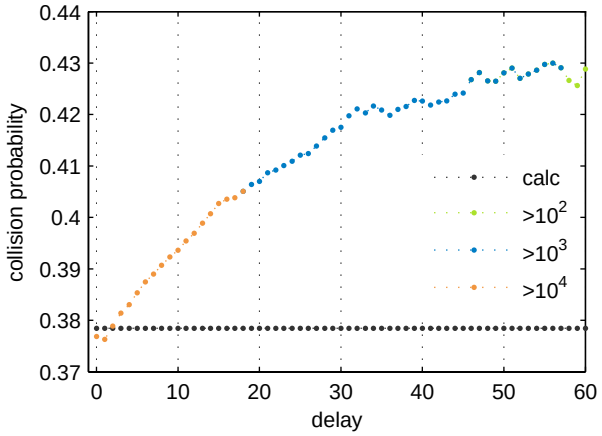


Figure 4.3. Shows the collision probability subject to the delay when using the flat model described in section 4.3.1. The colored dots show the simulation results, where the orange,blue and green dots are the average of respectively at least 10^4,10^3 and 10^2 simulation runs. The parameters were $N = 20$, $q = 0.1, \mathcal{D} = 10$ and $\lambda = 0.02$

Flat

The simplest construction of q is the constant, therefore independently of the postponement of the packets to be sent, sending probability:

$$p_k = \begin{cases} 0 & k = -1 \\ q & \text{otherwise} \end{cases} \tag{4.18}$$

REMARK: The value of q_{-1} is just a matter of form. Since no packet has to be sent, q_{-1} is limited to the value 0. ◇

Figure 4.3 shows the probability a collision happens, when a particular user with delay d send a packet. The longer the delay, the higher the collision probability and the more the calculated results differ from the simulation result. While the deviation at a delay of 10 time slots is 4 percent, this value increases to about 13 percent. But since the probability that a packet is delayed by a given amount of time decreases for bigger values, the growing deviation does not harm.

Peaks

Since the goal in this section is to minimize the delay respecting a deadline $\mathcal{D}$ (equations (4.16) and (4.17)), an obvious behavior of all players would be sending at the very last minute. In contrast to the previous, flat model, players at the states $k \ll \mathcal{D}$ and who have for this reason no costs awarded against themselfs do not send in order to keep the traffic low. On the other hand, participants who are close to pass the deadline do everything in their power to transmit a packet. That means only one thing: sending with probability 1. When the last minute effort didn't help, the participants tries to transmit the packet with constant probability q to stop the delay, and at the same time the costs, growing (model (4.16)) or only to be credited with a unit income.

The number of times, a user sends for sure before the deadline is represented by $\psi \in \mathbb{R}$. The fractional part of ψ (denoted by $\text{frac}(\psi) = \psi - \lfloor\psi\rfloor$) is interpreted as the sending probability before the real peaks begin:

$$p_k = \begin{cases} 0 & \text{otherwise} \\ \text{frac}(\psi) & k = \mathcal{D} - \lfloor\psi\rfloor \\ 1 & \mathcal{D} - \lfloor\psi\rfloor < k \leq \mathcal{D} \\ q & k > \mathcal{D} \end{cases} \tag{4.19}$$

Figure 4.4 shows the collision probability P'_{col} for a given delay when q is peak distributed. As we can see, the collision probability is underestimated by the calculation as long as the delay is smaller than the deadline $\mathcal{D}$, as soon as the deadline is passed, the simulated P'_{col} is bigger than the calculated result. Figure 4.4 shows the collision probability multiplied by the corresponding sending probability q.

Exponential

In contrast to the previous choice q and instead of waiting and sending suddenly with probability 1 shortly before the deadline, a more gentle increase

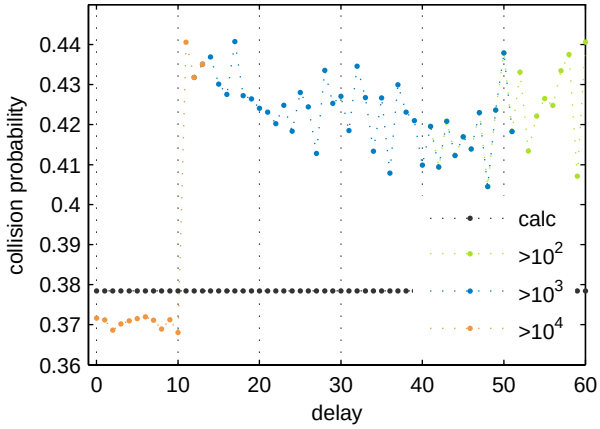


Figure 4.4. Shows the collision probability when using the peak model from section 4.3.1. The colored dots show the simulation results, where the orange,blue and green dots are the average of respectively at least 10^4,10^3 and 10^2 simulation runs. The parameters are $N = 20, q = 0.1, \mathcal{D} = 10$ and $\lambda = 0.02$

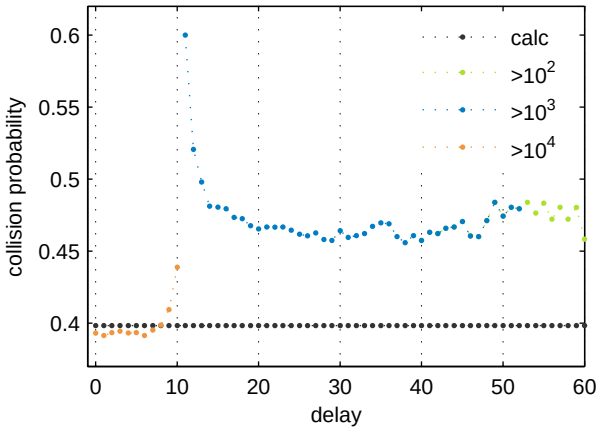


Figure 4.5. Shows the collision probability when using the exponential model from section 4.3.1. The colored dots show the simulation results, where the orange,blue and green dots are the average of respectively at least 10^4,10^3 and 10^2 simulation runs. The parameters are $N = 20, \beta = \alpha = 1, \mathcal{D} = 10$ and $\lambda = 0.02$

is chosen in this model:

$$p_k = \begin{cases} 0 & k = -1 \\ \alpha \cdot e^{-\beta(\mathcal{D}-k)} & 0 < k \leq \mathcal{D} \\ q & k > \mathcal{D} \end{cases} \tag{4.20}$$

Figure 4.5 shows the collision probability when $\alpha = 1$ and $\beta = 1$ and figure 4.6 shows the results when multiplying the values from figure 4.5 by the corresponding sending probability.

Figure 4.7 shows exemplary sending probabilities for the flat, exponential and peak model.

4.3.2 Evaluation

In this section we evaluate the models derived in the previous section. We plot the costs according to the two models (4.16) and (4.17) and compare them with the income. For each sending probability we plot an exemplary weighting γ of the income and the result. Since this weighting is arbitrary, we also calculate the bounds where the best result lies when considering every possible value γ in equation (4.15).

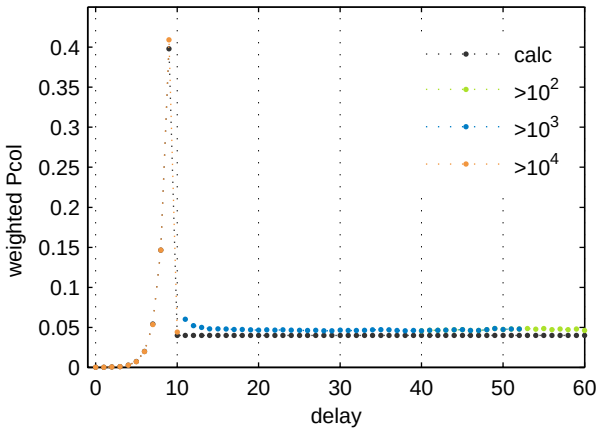


Figure 4.6. When multiplying the collision probabilities from figure 4.5 with the corresponding sending probability the values shown in this figure result.

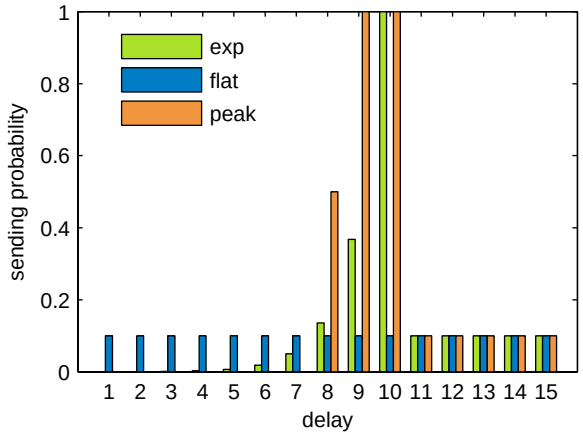


Figure 4.7. Three exemplary sending probability distributions when using the flat, the exponential and the peak model respectively. The deadline is $\mathcal{D}$ is equal to 10.

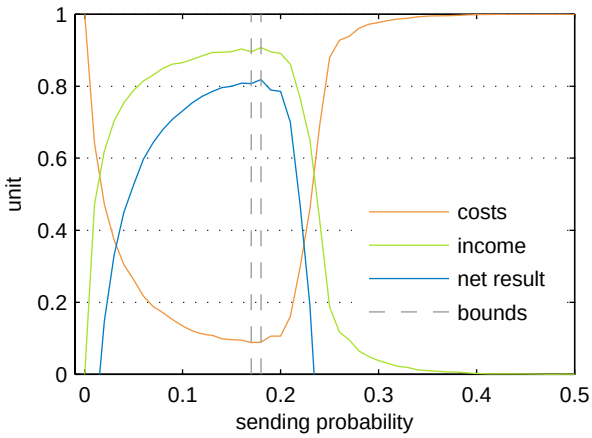


Figure 4.8. Shows the costs, the income (=throughput multiplied by 100) and the result (costs-income) when considering the costs model (4.16). The two gray broken lines show the bounds explained on page 64.

Flat model

First cost model Figure 4.8 shows the cost, income and gain (net result) when $\gamma = 100$. As you can see, the best result is reached for $q \in [0.17, 0.18]$.

Second cost model When we use the cost model (4.17) instead of (4.16), the set of optimal values q is increased and now ranges from 0 to 0.22.

Delay Finally we consider the bounds, such that 95%, 99% and 99.9% respectively of the delays that appear are smaller. For example, the 99%-bound at $q = 0.14$ is equal to 64. This means, in 99% percent of all cases a packet is delayed by at most 64 time slots.

The gray lines show the best q in terms of the bounds 95%, 99% and 99.9% respectivey. As you can see, these optima are between $q = 0.13$ and $q = 0.17$.

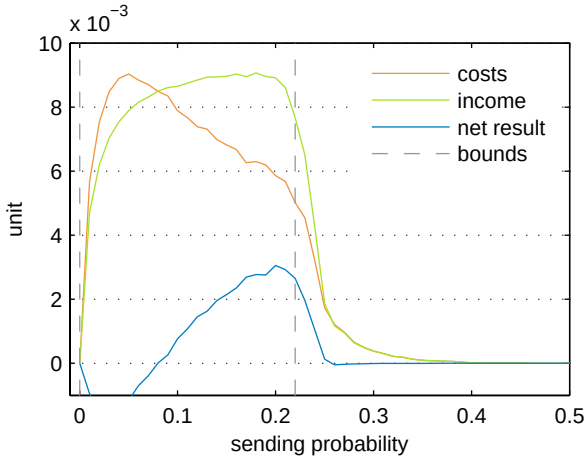


Figure 4.9. Shows the costs, income and net result when using the same parameters as in figure 4.8 and only changing the cost model to (4.17). In contrast to figure 4.8, the set of optimal values is bigger and ranges from 0 to 0.22.

Peak model

First cost model The cost according to the model (4.16), the income using $\gamma = 20$ and the net re-

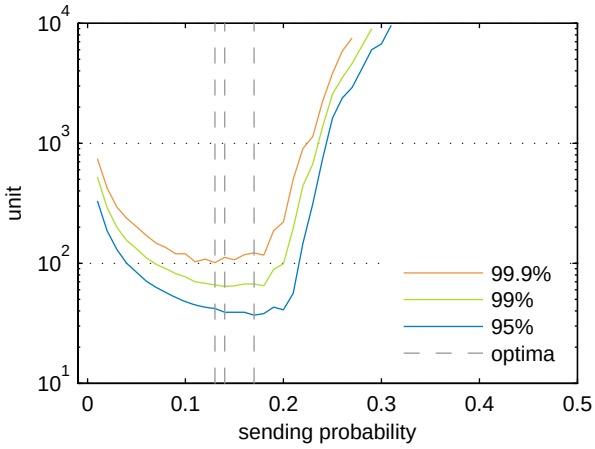


Figure 4.10. Shows the bounds, such that 95%, 99% and 99.9% respectively of the delays are smaller. The smaller the bound the better - the minima are marked by gray dashed lines.

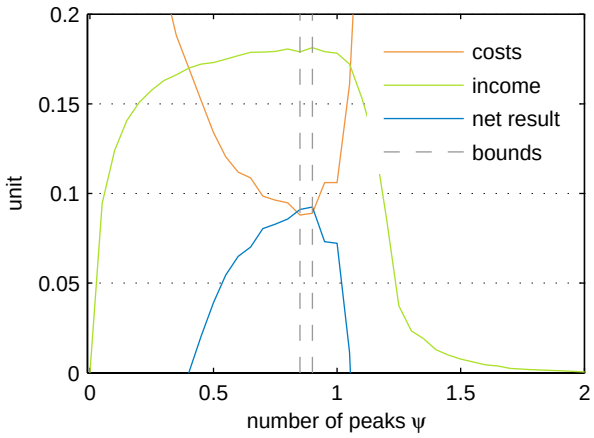


Figure 4.11. Shows the cost according to the model 4.16, the income using $\gamma = 20$ and the net result. The parameter q was set to 0.2. The best choice of ψ lies between 0.85 and 0.9 as the dashed gray lines indicate.

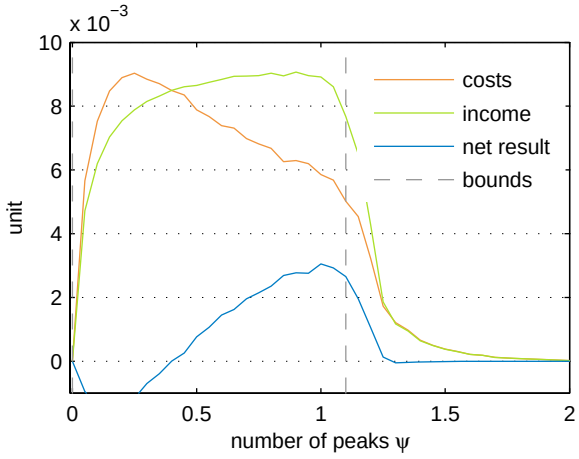


Figure 4.12. Costs according to model 4.17, income using $\gamma = 20$ and the net result for the peak model.

sult for a peak model is shown in figure 4.11. The parameter q was set to 0.2. As you can see, the optimal choice ψ lies between 0.85 and 0.9. That means, only one peak should be used.

Second cost model Changing the cost model (4.17) to (4.16) gives the results plotted in figure 4.12. This time, the optimal ψ ranges from 0 (no peak at all) to 1.1 (one "'complete"' peak plus another small peak of height 0.1).

Delay As for the flat case (figure 4.10) we consider the 95%-, 99%- and 99.9%-bounds respectively. As you can see, the optimal values marked by dashed gray lines lie between $\psi = 0.7$ and $\psi = 0.85$.

Exponential model

Since the exponential model uses two parameters α and β besides the probability q, we only modified β and setting α according to:

$$\alpha = \frac{1 - e^{-\beta \cdot \mathcal{D}}}{1 - e^{-\beta}} \tag{4.21}$$

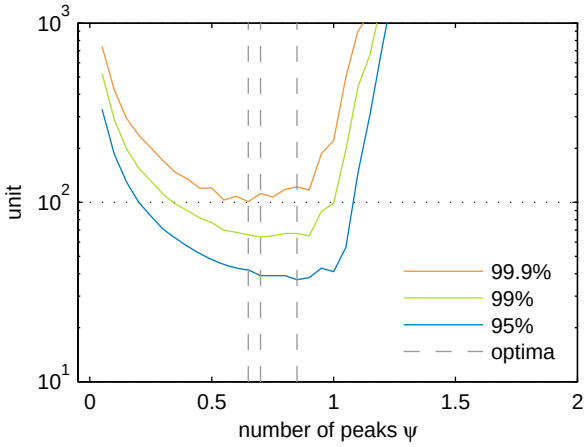


Figure 4.13. The 95%-, 99%- and 99.9%-delay-bounds. As you can see, the optimal values (marked by dashed gray lines) lie between $\psi = 0.7$ and $\psi = 0.85$.

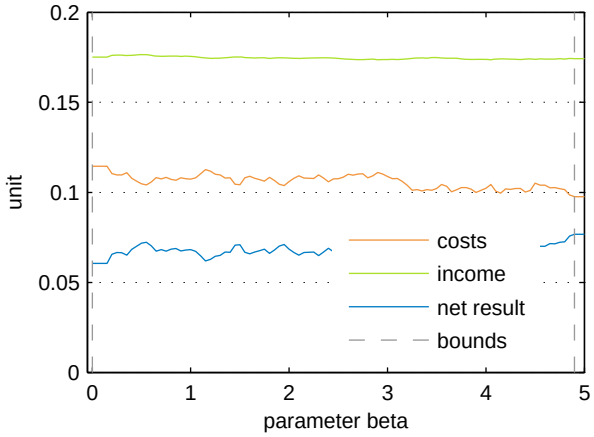


Figure 4.14. Shows the costs, the income (=throughput multiplied by 20) and the result (=costs-income) when considering the cost model (4.16). The two gray broken lines show the bounds explained on page 64.

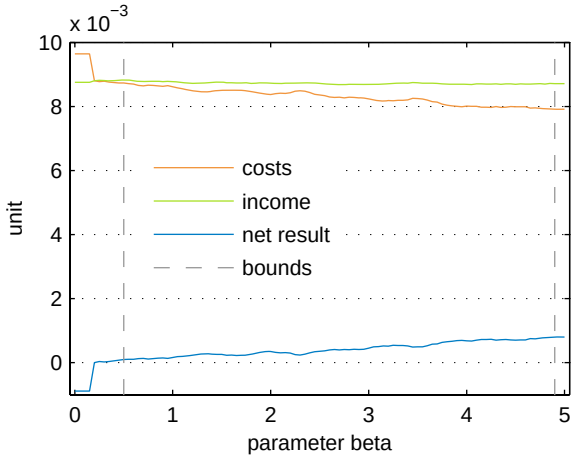


Figure 4.15. Shows the costs, income and net result when using the same parameters as in figure 4.14 and only changing the cost model to (4.17). The bigger β, the smaller the costs but at the same time the smaller the income.

First cost model Figure 4.16 shows the cost according to the model (4.16), the income using $\gamma = 20$ and the net result for various exponential models. The parameter q was set to 0.2. The costs and the income (and therefore the net result) nearly remain at the same level. The range of optimal parameters β covers the whole spectrum 0 to 4.9.

Second cost model When using cost model (4.17) instead of (4.16) we get the results plotted in figure 4.15. As for the first model (figure 4.14), the values don't change much. The costs slightly decrease for bigger values of β as well as the throughput. The optimal β ranges from 0.5 to 4.9.

Delay When looking at the 95%-, 99%- and 99.9%-bounds respectively we get the same diffuse results as for the cost models (figure 4.14 and 4.15 respectively). The optima are randomly distributed due to the noise of the simulation results.

4.3.3 Summary

The figures 4.17 and 4.18 show the cumulative distribution of the delay, using the flat, the expo-

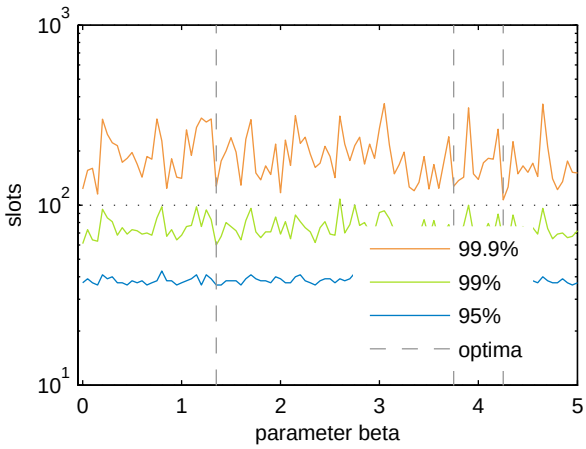


Figure 4.16. Shows the bounds, such that in 95%, 99% and 99.9% respectively of all cases the delay are of smaller value. Therefore smaller the bound the better. The minima are marked by gray dashed lines.

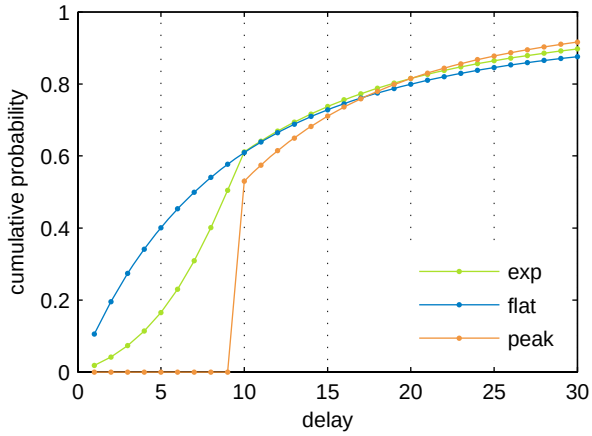


Figure 4.17. Shows the cumulative distribution of the delay for the three models flat, exponential and peak when they use the parameters obtained in section 4.3.2. As you can see, the course of the flat model increases the fastest in the beginning, but the exponential and the peak model catch up with the flat model and outperform it.

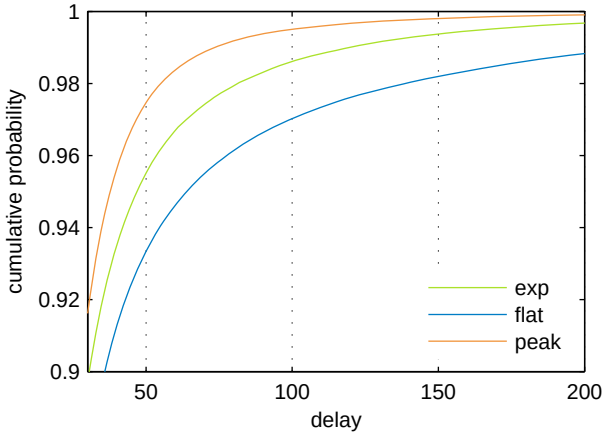


Figure 4.18. Shows the same cumulative distribution as in figure 4.17 for later time instances. As you can see, the peak model beats the exponential model, who for its part beats the flat sending probability distribution.

nential and the peak model with the parameters obtained in section 4.3.2. As you can see, after 10 time slots, the flat and the exponential model both have the same cumulated probability, while the peak cannot catch up with the big jump at the time instance 10. Around the time slot number 18, the peak distribution gains the lead in terms of cumulative probability and beats the exponential model, who for his part beats the flat model (see figure 4.18).

First cost model Figure 4.19 shows the costs according to equation (4.16) on page 62 where the best choices of the flat, peak and exponential distribution of q are used. The income is weighted by the factor 20. As you can see, the flat model performs worst in terms of throughput (and therefore income) as well as costs - the costs using the exponential or peak distribution are considerably smaller. The peak model beats the exponential model in terms of costs and provides the best result despite the slightly lower income.

Second cost model When we consider the second cost model explained in equation (4.17) on page

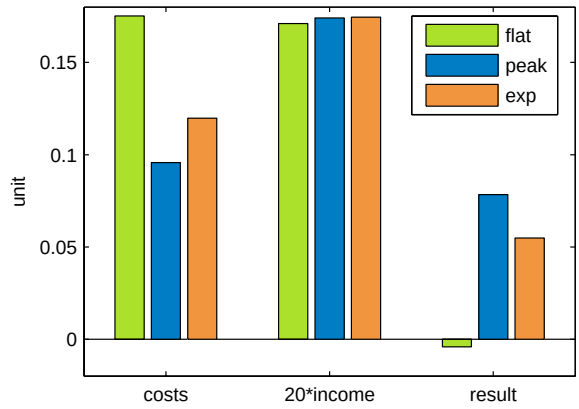


Figure 4.19. Shows the costs according to the model (4.16) on page 62. The results are determined by averaging a hundred simulation runs. The flat model uses $p = 0.21$, the peak model $\psi = 0.9, p = 0.2$ and the exponential model $\alpha = 0.5,\ \beta = 0.3,\ p = 0.2$. The income is scaled by a factor of 20, and the result is the calculated as follows: result = 20·income-costs.

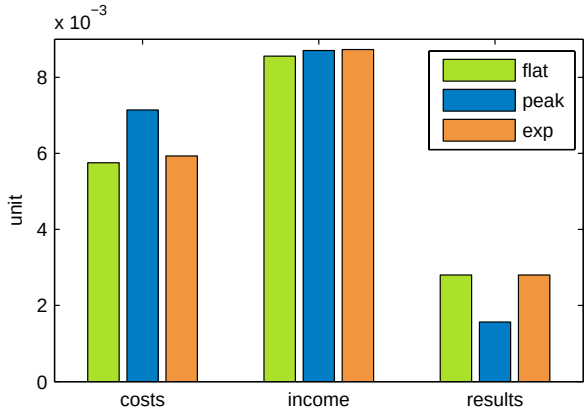


Figure 4.20. Shows the result when using the same parameters as in figure 4.19 and only changing the cost model to equation 4.17 on page 62. The result represents the difference between the income and the costs.

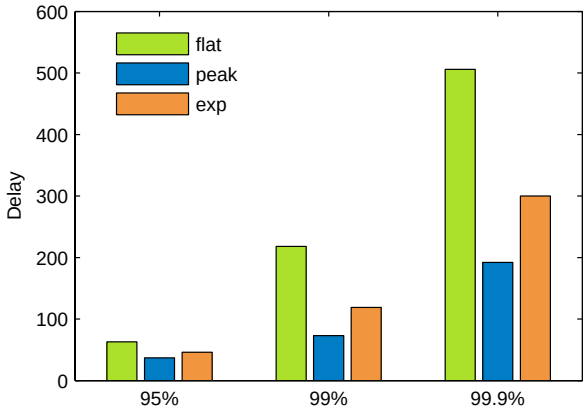


Figure 4.21. 95%, 99% and 99.9% respectively of the delay that appear are smaller than the bounds shown in this figure when using the same sending distributions as in figure 4.19.

62, the income stay the same since the cost model does not affect the throughput. The costs, however, are now maximal for the peak model. The smallest costs are reached by the flat model which gives also the best result, even though he only just wins with a result of $2.803 \cdot 10^{-3}$ chased by the exponential model who provides $2.798 \cdot 10^{-3}$.

Delay In figure 4.21 we consider the delay bound, such that 95%, 99% and 99.9% of the cases have a smaller delay. As you can see, the peak model beats the exponential model, who for his part beats the flat sending probability distribution.

Conclusions and Summary

In the first chapter (1.2.3) we derived a model to approximate the characteristics of the ALOHA-protocol - such as delay and throughput - when all participants use buffers to store packets incoming at a given rate λ. In contrast to most examples in the literature (for example [vM06]) the model is able to cope with every choice $\mathcal{B}$ of buffer space available and is not restricted to the case $\mathcal{B} = 1$. It was shown that the rather complex calculations involving a time dependent Markov chain can be reduced to the much simpler model all users sending with a probability determined only by the arrival rate of the packets.

In section 1.4 the optimal sending probability of unfair players among fair players was shown and in section 1.4.2 we gave bounds for these probabilities that guarantee the unfair users are not detected to a certain degree. As we have seen, the optimal behavior of unfair users does neither depend on the number nor the sending probability of the fair competitors but only on the number of unfair players. Since we excluded communication among unfair users to reach an agreement about their number, we showed how this quantity can be estimated counting the number of collisions (section 1.5).

As a second possibility to reach the optimal sending probability of unfair players we examined an adaptive version of ALOHA (chapter 2). But all configurations we considered turned out to result either in cannibalism (section 2) or a blocking condition (section 2.3.3).

In chapter 3 we introduced two methods to improve the throughput of the conventional ALOHA protocol simply by choosing the sending probability dependend on the delay of the packet to be transmitted. The approximation of the course of the characteristics - throughput and delay - of both protocols needed complex calculations involving Markov chains, but turned out to be very accurate as shown in section 3.3.2 and 3.3.3 respectively. The two protocols are able to increase the throughput (and decrease the delay) heavily considering long term behavior compared to the conventional ALOHA-protocol (sections 3.3.1 and 3.3.4). A crucial parameter of the two protocols turned out to be the period-length P, as explored in detail in section 3.4. Since both protocols don't need a setup phase or agreements among players they can be employed by unfair users and enhance their throughput considerably (section 3.5).

In the last chapter 4 we showed the impact of various cost models and derived the best sending probabilities strategies when focusing on three promising possibilities.

Bibliography

[Bor99] Jürgen Bortz. *Statistik für Sozialwissenschaftler*. Springer, 5th edition, 1999.

[GS03] Charles M. Grinstead and J. Laurie Snell. *Introduction to probability*, chapter Markov Chains, page 405 to 470. American Mathematical Society, 2nd edition, 2003.

[Mey00] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial & Applied Mathematics, 2000.

[Osb04] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2004.

[Pap01] Lothar Papula. volume 1. vieweg, 10th edition, 2001.

[Pou99] Alexander D. Poularikas. *The Handbook of Formulas and Tables for Signal Processing*, chapter Probability and Stochastic Processes. CRC Press, 1999.

[Pro01] John G. Proakis. *Digital Communications*. McGraw-Hill, New York, NY, USA, 4th international edition, 2001.

[Tan03] Andrew S. Tanenbaum. *Computer Networks*. Pearson Education International, 5th international edition, 2003.

[vM06] Piet van Mieghem. *Performance Analysis of Communications Networks and Systems*. Cambridge University Press, 2006.