WWW.PEARSON-STUDIUM.DE
CWS
COMPANION WEBSITE

Andrew S. Tanenbaum
David J. Wetherall

Computernetzwerke

5., aktualisierte Auflage

ALWAYS LEARNING
PEARSON

Andrew S. Tanenbaum
David J. Wetherall

Computernetzwerke

5., aktualisierte Auflage

PEARSON

Higher Education
München • Harlow • Amsterdam • Madrid • Boston
San Francisco • Don Mills • Mexico City • Sydney
a part of Pearson plc worldwide

▶Abbildung 4.1 zeigt die prinzipielle Darstellung der Erzeugung von Rahmen in einem ALOHA-System. Die Rahmen haben alle die gleiche Länge, da der Datendurchsatz von ALOHA-Systemen dadurch maximiert wird, dass eine feste Rahmengröße vorgeschrieben ist.

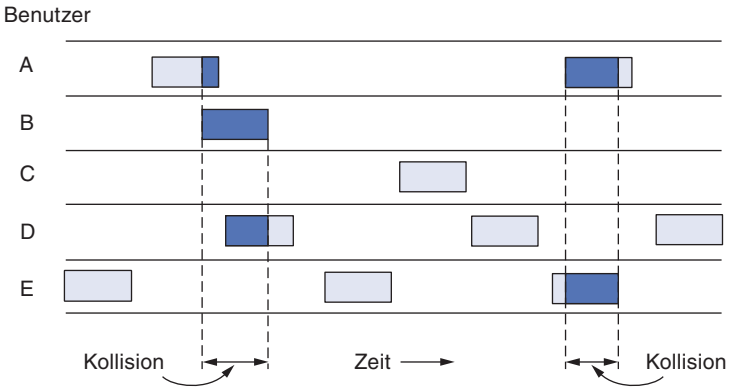

Abbildung 4.1: Beim reinen ALOHA werden Rahmen zu beliebigen Zeiten übertragen.

Jedes Mal, wenn zwei Rahmen zur gleichen Zeit versuchen, den Kanal zu besetzen, entsteht eine Kollision (siehe Abbildung 4.1) und beide werden verstümmelt. Falls auch nur das erste Bit eines neuen Rahmens das letzte Bit eines fast beendeten Rahmens überschneidet, werden beide Rahmen völlig zerstört (d.h. haben falsche Prüfsummen) und müssen später erneut übertragen werden. Eine Prüfsumme kann nicht (und sollte auch nicht) zwischen einem totalen und einem teilweisen Verlust unterscheiden.

Eine interessante Frage ist: Wie sieht die Effizienz eines ALOHA-Systems aus? Mit anderen Worten, welcher Teil aller übermittelten Rahmen entgeht unter diesen chaotischen Bedingungen einer Kollision? Betrachten wir zunächst eine unendliche Menge von Benutzern, die etwas in ihre Terminals (Stationen) eingeben. Ein Benutzer befindet sich immer in einem von zwei Zuständen: Er schreibt oder er wartet. Zu Beginn befinden sich alle Benutzer im schreibenden Zustand. Wenn eine Zeile fertig geschrieben ist, hört der Benutzer zu schreiben auf und wartet auf eine Antwort. Dann überträgt die Station einen Rahmen, in dem sich die Zeile befindet, über den gemeinsamen Kanal zum Zentralrechner und prüft den Kanal, um zu sehen, ob die Übertragung erfolgreich war. Trifft das zu, sieht der Benutzer die Antwort und fährt mit dem Schreiben fort. Andernfalls wartet der Benutzer, während die Station den Rahmen immer wieder neu überträgt, bis er erfolgreich angekommen ist.

Die Rahmenübertragungszeit sei die Zeit, die benötigt wird, um den Standardrahmen mit fester Länge zu übermitteln (d.h. die Länge des Rahmens geteilt durch die Bitübertragungsrate). Wir gehen hier davon aus, dass neue Rahmen, die von den Stationen erzeugt werden, durch eine Poisson-Verteilung mit einer mittleren Anzahl von N Rahmen pro Rahmenübertragungszeit wohlmodelliert sind. (Die Annahme einer unbegrenzten Benutzerzahl ist erforderlich, um sicherzustellen, dass N nicht abnimmt, wenn die Benutzer gerade blockiert sind.) Falls $N > 1$ ist, erzeugen die Benutzer die

Rahmen schneller als der Kanal sie verarbeiten kann, also erleidet fast jeder Rahmen eine Kollision. Für einen vernünftigen Durchsatz gehen wir von $0 < N < 1$ aus.

Abgesehen von den neuen Rahmen generieren die Stationen auch Neuübertragungen von vorher verunglückten Rahmen. Wir nehmen an, dass die alten und die neuen Rahmen zusammen durch eine Poisson-Verteilung wohlmodelliert sind, mit einem mittleren Wert von G Rahmen pro Rahmenübertragungszeit. Natürlich gilt $G \geq N$. Bei niedriger Belastung (d.h. $N \approx 0$) gibt es wenig Kollisionen, also auch wenig Neuübertragungen, sodass $G \approx N$ ist. Bei höheren Lasten entstehen viele Kollisionen, sodass $G > N$ ist. Für jede beliebige Last ist der Datendurchsatz S die gegebene Last G multipliziert mit der Wahrscheinlichkeit, dass eine Übertragung erfolgreich ist, d.h. $S = GP_0$, wobei P_0 die Wahrscheinlichkeit ist, dass ein Rahmen keine Kollision erleidet.

Ein Rahmen kollidiert nicht, wenn keine anderen Rahmen innerhalb einer Rahmenübertragungszeit nach seinem Start gesendet werden (▶Abbildung 4.2). Unter welchen Bedingungen kommt der graue Rahmen unbeschädigt an? Angenommen, t ist die zum Senden eines Rahmens benötigte Zeit. Wenn ein anderer Benutzer in der Zeit zwischen t_0 und t_0+t einen Rahmen erstellt, kollidiert das Ende dieses Rahmens mit dem Anfang des grauen Rahmens. In Wirklichkeit war das Schicksal des grauen Rahmens schon vor der Übertragung des ersten Bits besiegelt. Da beim reinen ALOHA eine Station den Kanal vor der Übertragung nicht prüft, hat sie keine Möglichkeit zu erfahren, dass ein anderer Rahmen schon unterwegs ist. Ebenso wird jeder andere Rahmen, der zwischen t_0+t und t_0+2t abgesendet wird, mit dem Ende des dunkelblauen Rahmens zusammenstoßen.

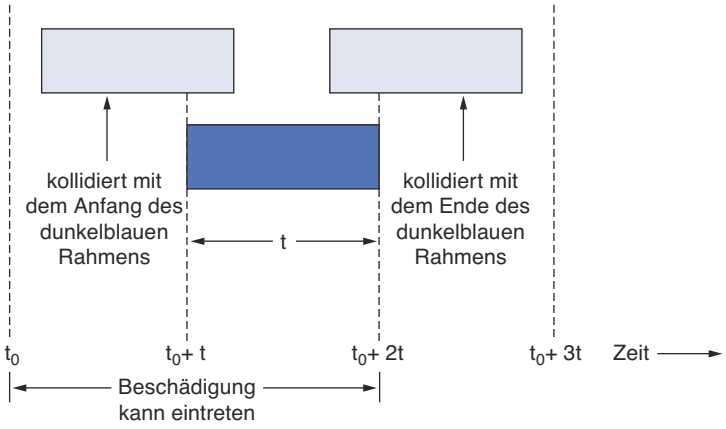

Abbildung 4.2: Gefährliche Zeitspanne für den dunkelblauen Rahmen.

Die Wahrscheinlichkeit, dass k Rahmen innerhalb einer gegebenen Rahmenübertragungszeit produziert werden, in der G Rahmen erwartet werden, ist durch die Poisson-Verteilung

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \tag{4.2}$$

gegeben. Die Wahrscheinlichkeit für 0 Rahmen ist daher genau e^{-G}. Innerhalb einer Zeitspanne, die zwei Rahmenübertragungszeiten lang ist, werden im Durchschnitt $2G$ Rahmen erzeugt. Die Wahrscheinlichkeit, dass innerhalb der gefährlichen Zeitspanne keine weiteren Rahmen begonnen werden, ist somit $P_0 = e^{-2G}$. Unter Verwendung von $S = GP_0$ erhalten wir

$$S = Ge^{-2G}$$

Das Verhältnis zwischen dem gegebenen Datenverkehr und dem Datendurchsatz wird in ▶Abbildung 4.3 dargestellt. Der maximale Durchsatz erfolgt bei $G = 0{,}5$ mit $S = 1/2e$, was etwa 0,184 ergibt. Mit anderen Worten: Wir können bestenfalls mit einer Kanalnutzung von maximal 18 % rechnen. Dieses Ergebnis ist nicht sehr ermutigend. Wenn aber jeder Daten dann überträgt, wann er will, ist kaum eine Erfolgsrate von 100 % zu erwarten.

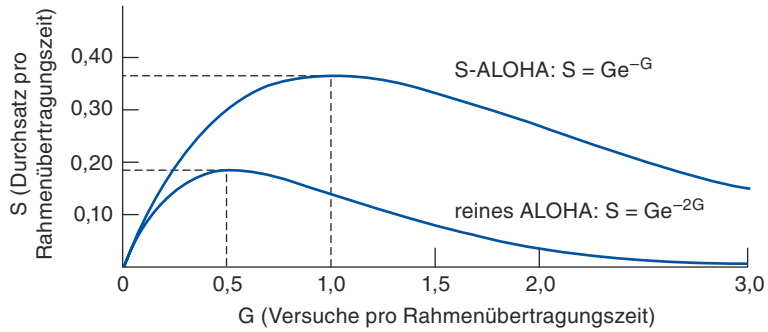

Abbildung 4.3: Durchsatz gegenüber angebotenem Datenverkehr bei ALOHA-Systemen.

S-ALOHA

Kurz nachdem ALOHA in der Szene auftauchte, veröffentlichte Roberts (1972) eine Methode zur Verdoppelung der Kapazität eines ALOHA-Systems. Er schlug vor, die Zeit in einzelne Intervalle – sogenannte **Zeitscheiben** (*slot*) – aufzuteilen, wobei jedes Intervall einem Rahmen entspricht. Bei diesem Ansatz müssen sich die Benutzer bezüglich der Grenzen der Zeitscheiben abstimmen. Eine Möglichkeit, eine gewisse Synchronisation zwischen den Benutzern zu erreichen, wäre die, dass eine Station zu Beginn eines jeden Intervalls wie eine Uhr einen Piepton aussendet.

Bei der Methode von Roberts, die als **S-ALOHA** (Slotted ALOHA) bekannt wurde, darf im Gegensatz zum **reinem ALOHA** von Abramson eine Station nicht sofort senden, wenn der Benutzer eine Zeile eingibt. Stattdessen muss auf die nächste Zeitscheibe gewartet werden. So wird der zeitlich ununterbrochene Fluss des reinen ALOHA in eine Variante mit diskreten Zeitabschnitten abgewandelt. Dies halbiert die gefährliche Zeitspanne. Betrachten Sie dazu Abbildung 4.3 und überlegen Sie sich die nun möglichen Kollisionen. Die Wahrscheinlichkeit, dass kein anderer Datenverkehr in der gleichen Zeitscheibe wie unser Testrahmen stattfindet, ist e^{-G}, woraus sich

$$S = Ge^{-G} \tag{4.3}$$

ergibt. Wie man in Gleichung (4.3) sieht, hat S-ALOHA seinen Spitzenwert bei $G=1$ mit einem Durchsatz von $S=1/e$ bzw. ungefähr 0,368. Dies ist also eine Verdoppelung gegenüber dem reinen ALOHA. Wenn das System mit $G=1$ arbeitet, ist die Wahrscheinlichkeit für eine leere Zeitscheibe 0,368 (aus Gleichung (4.2)). Im besten Fall liefert S-ALOHA 37 % leere Zeitscheiben, 37 % erfolgreiche Übertragungen und 26 % Kollisionen. Ein Betrieb mit höheren Werten für G verringert zwar die Zahl der leeren Zeitscheiben, erhöht aber die Zahl der Kollisionen exponentiell. Um zu verstehen, wie dieser schnelle Zuwachs von Kollisionen mit G zusammenhängt, betrachten wir als Beispiel die Übertragung eines Testrahmens. Die Wahrscheinlichkeit, dass er eine Kollision vermeiden kann, ist e^{-G}, welches die Wahrscheinlichkeit ist, dass alle anderen Stationen nicht in der gleichen Zeitscheibe übertragen. In diesem Fall beträgt die Wahrscheinlichkeit einer Kollision nur $1-e^{-G}$. Die Wahrscheinlichkeit, dass eine Übertragung genau k Versuche benötigt (d.h. $k-1$ Kollisionen, gefolgt von einem Erfolg) ist

$$P_k = e^{-G}(1-e^{-G})^{k-1}$$

Die erwartete Anzahl an Übertragungen E pro am Terminal eingegebener Zeile ist dann

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1-e^{-G})^{k-1} = e^{G}$$

Als Folge der exponentiellen Abhängigkeit von E von G können bereits geringe Erhöhungen der Kanallast die Leistung erheblich verringern.

S-ALOHA ist darüber hinaus aus einem Grund bemerkenswert, der nicht auf den ersten Blick ersichtlich ist. Es wurde in den 1970er Jahren entwickelt, in ein paar experimentellen Systemen eingesetzt und geriet dann überwiegend in Vergessenheit. Als der Internetzugang über Kabel erfunden wurde, entstand plötzlich das Problem, wie man einen gemeinsam genutzten Kanal unter mehreren konkurrierenden Benutzern zuweisen sollte. Hier wurde dann S-ALOHA zur Rettung der Lage wieder aus der Versenkung geholt. Die Situation, dass mehrere RFID-Tags mit demselben RFID-Lesegerät kommunizieren wollen, stellt eine weitere Variation desselben Problems dar. S-ALOHA, gemischt mit einem Spritzer weiterer Ideen, wurde erneut zur Rettung herangezogen. Es ist schon öfter vorgekommen, dass vollkommen gültige Protokolle aus politischen Gründen nicht mehr verwendet werden (beispielsweise weil ein großes Unternehmen möchte, dass alle nach seinem Verfahren arbeiten) oder aufgrund von sich ständig verändernden Technologietrends. Doch dann, Jahre später, erkennt jemand, dass das eigentlich vergessene Protokoll sein aktuelles Problem löst. Aus diesem Grunde beschäftigen wir uns in diesem Kapitel mit mehreren eleganten Protokollen, die zwar zurzeit keine große Verbreitung aufweisen, aber vielleicht in zukünftigen Anwendungen einen Einsatz finden – vorausgesetzt, dass viele Netzentwickler sie kennen. Natürlich behandeln wir auch viele Protokolle, die aktuell benutzt werden.

4.2.2 CSMA-Protokolle (Carrier Sense Multiple Access)

Bei S-ALOHA liegt die bestmögliche Kanalauslastung bei 1/*e*. Dieses niedrige Ergebnis ist nicht erstaunlich, da hier die Stationen senden, wann sie wollen, und die Aktivitäten der anderen Stationen nicht kennen, sodass viele Kollisionen zu erwarten sind. In LANs können die Stationen demgegenüber häufig erkennen, was andere Stationen tun, und somit ihr eigenes Verhalten anpassen. Diese Netze können eine viel bessere Auslastung als 1/*e* erreichen. In diesem Abschnitt werden einige Protokolle mit Hinblick auf die Leistungsverbesserung behandelt.

Protokolle, bei denen Stationen einen Träger (d.h. eine Übertragung) abhören und dementsprechend handeln, heißen **Protokolle mit Trägerprüfung** (*carrier sense protocol*). Hier wurden inzwischen mehrere Varianten vorgeschlagen und schon vor langer Zeit detailliert analysiert, siehe zum Beispiel Kleinrock und Tobagi (1975). Im Folgenden werden wir uns verschiedene Varianten von Protokollen mit Trägerprüfung ansehen.

Persistentes CSMA und nicht persistentes CSMA

Als erstes Protokoll mit Trägerprüfung untersuchen wir das **1-persistente CSMA** (*Carrier Sense Multiple Access*, Mehrfachzugriff mit Trägerprüfung). Dieser Name ist ein wenig sperrig für das einfachste CSMA-Verfahren. Wenn eine sendewillige Station Daten übertragen möchte, hört sie erst den Kanal ab, ob bereits jemand sendet. Ist der Kanal frei, dann sendet die Station ihre Daten. Wenn der Kanal dagegen besetzt ist, so wartet die Station einfach, bis er frei wird, dann überträgt sie einen Rahmen. Falls eine Kollision auftritt, wartet die Station eine zufällige Zeitspanne und beginnt von vorn. Dieses Protokoll heißt 1-persistent, weil die Station mit einer Wahrscheinlichkeit von 1 sendet, wenn der Kanal im Leerlaufzustand ist.

Man könnte nun erwarten, dass dieses Verfahren Kollisionen bis auf den seltenen Fall von gleichzeitigem Senden vermeidet, doch tatsächlich tut es das nicht. Wenn zwei Stationen sendebereit werden, während gerade eine dritte Station sendet, so warten beide höflich, bis diese Übertragung beendet ist, und dann beginnen beide genau gleichzeitig zu senden, was zu einer Kollision führt. Wenn sie nicht so ungeduldig wären, gäbe es weniger Kollisionen.

Außerdem hat auch die Ausbreitungsverzögerung einen großen Einfluss auf Kollisionen. Es besteht die Möglichkeit, dass kurz nachdem eine Station zu senden begonnen hat, eine andere Station auch senden will und den Kanal überprüft. Wenn das Signal der ersten Station die zweite noch nicht erreicht hat, findet die zweite einen Kanal im Leerlaufzustand vor und beginnt ebenfalls zu senden, was zu einer Kollision führt. Diese Möglichkeit hängt von der Anzahl der Rahmen ab, die sich auf dem Kanal aufhalten können, bzw. dem **Bandbreite-Verzögerung-Produkt** des Kanals. Wenn nur ein kleiner Teil eines Rahmens auf den Kanal passt, wie dies für die meisten LANs der Fall ist, da hier die Ausbreitungsverzögerung gering ist, dann tritt selten eine Kollision auf. Je größer das Bandbreite-Verzögerung-Produkt ist, umso wichtiger wird dieser Effekt und umso geringer wird die Leistungsfähigkeit des Protokolls.

Trotz allem hat dieses Protokoll eine bessere Performanz als das reine ALOHA, da beide Stationen genug Anstand besitzen, nicht mit dem Rahmen der dritten Station in Konflikt zu geraten. Genau das Gleiche gilt auch für S-ALOHA.

Die zweite Variante eines Protokolls mit Trägerprüfung ist **nicht-persistentes CSMA**. Bei diesem Protokoll mit Trägerprüfung wird bewusst der Versuch unternommen, etwas weniger gierig zu sein als im vorherigen Protokoll. Wie vorher überprüft eine Station den Kanal, wenn es einen Rahmen senden möchte, und falls niemand anderes sendet, fängt die Station selbst damit an. Ist allerdings der Kanal bereits belegt, dann überprüft ihn die Station nicht andauernd mit dem Hintergedanken, ihn sofort an sich zu reißen, sobald das Ende der vorherigen Übertragung erkannt wird. Stattdessen wird eine zufällige Zeitspanne gewartet und der Vorgang dann wiederholt. Dieser Algorithmus führt zu einer besseren Kanalauslastung, aber längeren Wartezeiten als beim 1-persistenten CSMA.

Eine weitere Protokollvariante ist **p-persistentes CSMA**. Dieses Protokoll kann auf Kanäle mit Zeitscheiben angewandt werden und funktioniert wie folgt: Wenn eine Station senden will, überprüft sie den Kanal. Ist er frei, sendet die Station mit einer Wahrscheinlichkeit p. Mit einer Wahrscheinlichkeit von $q=1-p$ wartet sie bis zur nächsten Zeitscheibe. Ist diese Zeitscheibe auch frei, wird entweder gesendet oder gewartet, wiederum mit den Wahrscheinlichkeiten p und q. Dieser Vorgang wird wiederholt, bis entweder der Rahmen übertragen worden ist oder eine andere Station zu senden begonnen hat. Im letzteren Fall reagiert die glücklose Station, als ob eine Kollision stattgefunden hätte (d.h. eine zufällige Zeitspanne warten und dann alles von vorn beginnen). Wenn die Station erkennt, dass der Kanal belegt ist, wartet sie bis zur nächsten Zeitscheibe und wendet dann das obige Verfahren an. IEEE 802.11 benutzt eine Verfeinerung von p-persistentem CSMA, welches wir in Abschnitt 4.4 besprechen.

▶Abbildung 4.4 zeigt den berechneten Durchsatz in Abhängigkeit vom anfallenden Datenverkehr für alle drei Protokolle sowie für reines ALOHA und S-ALOHA.

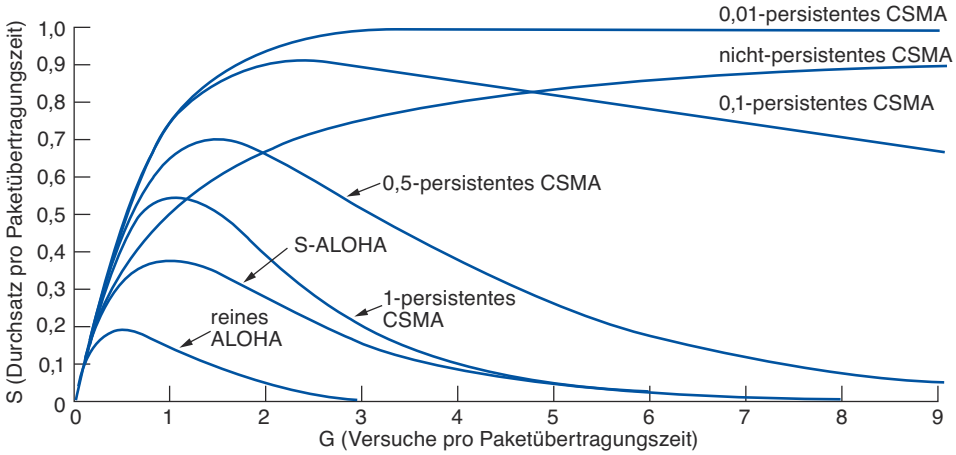

Abbildung 4.4: Vergleich der Kanalauslastung in Abhängigkeit vom Datenverkehr bei verschiedenen zufallsgesteuerten Protokollen.

CSMA mit Kollisionserkennung

Persistente und nicht persistente CSMA-Protokolle sind sicher eine Verbesserung gegenüber ALOHA, da sie gewährleisten, dass keine Station sendet, während der Kanal belegt ist. Wenn jedoch zwei Stationen den Kanal als frei erkennen und beide gleichzeitig zu senden beginnen, werden ihre Signale immer noch kollidieren. Eine weitere Verbesserung ist, dass die Station die Kollision schnell entdecken und die Übertragung abrupt abbrechen kann (statt sie normal zu beenden), da diese ohnehin irreparabel verstümmelt wäre. Diese Strategie spart Zeit und Bandbreite.

Dieses Protokoll heißt **CSMA/CD** (*Carrier Sense Multiple Access with Collision Detection*, Mehrfachzugang mit Trägerprüfung und Kollisionserkennung) und ist die Grundlage für das klassische Ethernet-LAN, sodass es hier durchaus eine nähere Betrachtung verdient. Es ist wichtig sich zu vergegenwärtigen, dass Kollisionserkennung ein analoger Prozess ist. Die Hardware der Station muss den Kanal während der Übertragung abhören. Wenn das Signal, das zurückkommt, sich von dem Signal unterscheidet, das ursprünglich auf den Kanal gelegt wurde, dann weiß die Station, dass eine Kollision aufgetreten ist. Dies impliziert, dass ein empfangenes Signal im Vergleich zum gesendeten Signal nicht winzig sein muss (was bei kabellosen Kanälen schwierig ist, da empfangene Signale 1 000 000-mal schwächer als gesendete Signale sein können) und dass die Modulation so gewählt werden muss, um Kollisionserkennung zu ermöglichen (z. B. könnte es gut sein, dass es unmöglich ist, eine Kollision von zwei 0-Volt-Signalen zu entdecken).

CSMA/CD verwendet genau wie viele andere LAN-Protokolle das Konzept aus ▶Abbildung 4.5. Zum Zeitpunkt t_0 hat eine Station die Übertragung ihres Rahmens beendet. Jede andere Station, die einen Rahmen zu senden hat, kann das jetzt versuchen. Wenn sich zwei oder mehr Stationen gleichzeitig zum Senden entschließen, kommt es zu einer Kollision. Wenn eine Station die Kollision erkennt, unterbricht sie ihre Übertragung, wartet eine zufällige Zeitspanne und versucht es dann erneut (unter der Annahme, dass inzwischen keine andere Station zu übertragen begonnen hat). Deshalb besteht unser CSMA/CD-Modell aus abwechselnden Konkurrenz- und Übertragungsperioden, wobei Leerlauf entsteht, sobald keine Station sendet.

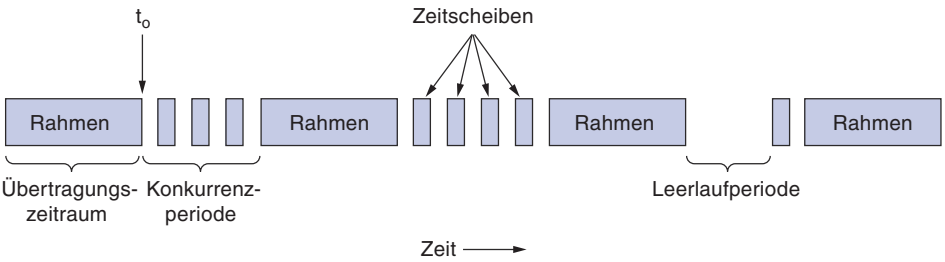

Abbildung 4.5: CSMA/CD kann sich im Konkurrenz-, Übertragungs- oder Leerlaufzustand befinden.

Wir wollen nun die Einzelheiten des Konkurrenzalgorithmus genauer untersuchen. Angenommen, zwei Stationen beginnen genau zur Zeit t_0 mit der Übertragung. Wie lange brauchen sie um zu erkennen, dass eine Kollision stattgefunden hat? Die Beant-

wortung dieser Frage ist grundlegend für die Ermittlung der Konkurrenzperiode, aus der Verzögerung und Datendurchsatz bestimmt werden.

Die minimale Zeit bis zur Entdeckung einer Kollision ist die Zeit, die ein Signal braucht, um von einer Station zur anderen zu gelangen. Aufgrund dieser Information könnte man folgern, dass eine Station, die seit Beginn der Übertragung für die Dauer der Signalausbreitung über das Kabel hinweg keine Kollision erkannt hat, sicher sein kann, sie verfüge über das Kabel. Mit „verfügen“ ist gemeint, dass alle anderen Stationen die Übertragung erkannt haben und demzufolge nicht stören. Diese Schlussfolgerung ist falsch.

Stellen Sie sich als schlechtesten Fall folgendes Szenario vor: Angenommen, τ ist die Zeit, die ein Signal für die Ausbreitung zwischen den beiden am weitesten entfernten Stationen benötigt. Bei t_0 beginnt eine Station mit der Übertragung. Zur Zeit $t_0+\tau-\varepsilon$, kurz bevor das Signal die entfernteste Station erreicht, beginnt auch diese zu senden. Natürlich entdeckt sie die Kollision fast im gleichen Augenblick und bricht ab, aber das kurze Signal, das durch die Kollision erzeugt wurde, gelangt zur ursprünglichen Station nicht vor der Zeit $2\tau-\varepsilon$ zurück. Das bedeutet, dass eine Station im schlechtesten Fall nicht sicher sein kann, den Kanal zu belegen, bevor sie 2τ lang gesendet hat, ohne eine Kollision festzustellen.

Mit diesem Verständnis können wir uns CSMA/CD-Konkurrenz als S-ALOHA-System mit einer Zeitscheibengröße von 2τ vorstellen. In einem 1 km langen Koaxialkabel gilt $\tau \approx 5$ µs. Der Unterschied zwischen CSMA/CD und S-ALOHA ist, dass Zeitscheiben, in denen nur eine Station überträgt (d.h., in denen der Kanal belegt ist), vom Rest eines Rahmens gefolgt werden. Dieser Unterschied verbessert die Performanz bedeutend, falls die Rahmenzeit sehr viel länger als die Übertragungszeit ist.

4.2.3 Kollisionsfreie Protokolle

Obwohl Kollisionen bei CSMA/CD nicht mehr entstehen, nachdem eine Station eindeutig den Kanal in Besitz hält, können sie während der Konkurrenzperiode immer noch auftreten. Diese Kollisionen beeinflussen die Leistungsfähigkeit des Systems im negativen Sinn, vor allem wenn das Bandbreite-Verzögerung-Produkt sehr groß ist, z.B. wenn das Kabel lang ist (d.h. ein großer τ-Wert) und kurze Rahmen übertragen werden. Kollisionen verringern nicht nur die Bandbreite, darüber hinaus bewirken sie, dass die Zeit zum Senden eines Rahmens variiert, was für Echtzeitdatenverkehr wie VoIP keine gute Eigenschaft ist. CSMA/CD ist außerdem nicht universell einsetzbar.

In diesem Abschnitt werden wir einige Protokolle untersuchen, die die Konkurrenz um den Kanal auf eine Art lösen, bei der keine Kollisionen auftreten, nicht einmal während der Konkurrenzperiode. Die meisten dieser Protokolle werden derzeit nicht in praktisch bedeutsamen Systemen verwendet. Aber in einem sich sehr schnell wandelnden Bereich ist es oft sehr vorteilhaft, Protokolle mit exzellenten Eigenschaften für zukünftige Systeme bereits an der Hand zu haben.

Bei allen beschriebenen Protokollen nehmen wir an, dass es N Stationen gibt, jede mit einer eindeutigen Adresse von 0 bis $N-1$ programmiert. Es spielt keine Rolle, dass einige Stationen ab und zu inaktiv sind. Wir gehen weiter davon aus, dass die Signalausbreitungsverzögerung vernachlässigbar ist. Die Hauptfrage bleibt: Welche Station erhält nach einer erfolgreichen Übertragung den Kanal? Wir benutzen weiterhin das Modell von Abbildung 4.5 mit den diskreten Zeitscheiben für konkurrierende Übertragungsversuche.

Bitmusterprotokoll

In unserem ersten kollisionsfreien Protokoll, der **Bitmuster-Methode** (*basic bit-map method*), besteht jede Konkurrenzperiode aus genau N Zeitscheiben. Wenn Station 0 einen Rahmen senden will, überträgt sie in Zeitscheibe 0 ein 1-Bit. Keine andere Station darf in dieser Zeitscheibe übertragen. Unabhängig davon, was Station 0 macht, erhält Station 1 die Möglichkeit, in Zeitscheibe 1 ein 1-Bit zu übertragen, aber nur wenn sie einen zur Übertragung bereitstehenden Rahmen hat. Allgemein kann Station j die Tatsache eines zur Übertragung anstehenden Rahmens dadurch anzeigen, dass sie in Zeitscheibe j eine 1 einfügt. Nachdem alle N Zeitscheiben durchlaufen sind, hat jede Station den genauen Überblick, welche Stationen übertragen wollen. Jetzt beginnen sie der Reihe nach Rahmen zu senden (▶Abbildung 4.6).

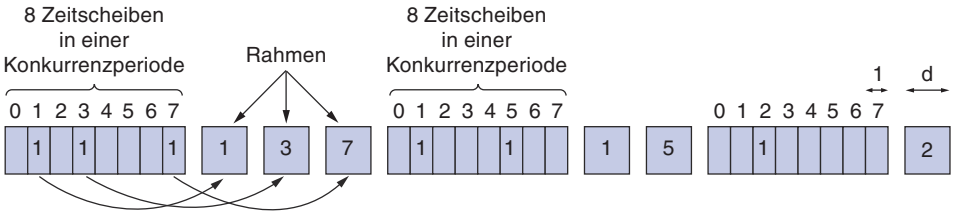

Abbildung 4.6: Das Bitmusterprotokoll.

Da sich bei jeder Übertragung alle darüber einigen, wer als Nächstes an die Reihe kommt, entstehen nie Kollisionen. Nachdem die letzte bereite Station ihren Rahmen übertragen hat, was alle Stationen leicht mitverfolgen können, beginnt eine weitere N-Bit-Konkurrenzperiode. Wenn eine Station erst kurz nach Ablauf ihrer Bitscheibe sendebereit wird, hat sie Pech gehabt und muss warten, bis jeder an der Reihe war und das Bitmuster wieder die Runde gemacht hat. Protokolle wie dieses, bei dem jeder Übertragungswunsch vor der tatsächlichen Übertragung allen Stationen bekannt gegeben wird, nennt man **Reservierungsprotokolle** (*reservation protocol*), weil sie Kanalbesitz im Voraus reservieren und Kollisionen verhindern. Nun wollen wir kurz die Leistungsfähigkeit dieses Protokolls analysieren. Der Einfachheit halber messen wir die Zeit in Einheiten der Größe einer Zeitscheibe in der Konkurrenzperiode, wobei Datenrahmen aus d Zeiteinheiten bestehen.

Im Zustand geringer Belastung wird einfach mangels Datenrahmen das Bitmuster immer wiederholt. Wir betrachten die Situation aus Sicht einer Station mit niedriger Nummer, z.B. 0 oder 1. Wenn sie senden will, wird die „aktuelle“ Zeitscheibe meist irgendwo in der Mitte des Bitmusters sein. Im Durchschnitt muss die Station $N/2$ Zeit-

scheiben warten, bis die aktuelle Abfrage vorbei ist, und weitere N Zeitscheiben bis zur Beendigung der nächsten Abfrage, bevor übertragen werden kann.

Die Aussichten für Stationen mit höheren Nummern sind besser. Normalerweise müssen diese, bevor sie mit einer Übertragung beginnen können, nur für die Dauer einer halben Prüfung ($N/2$ Bitscheiben) warten. Stationen mit höheren Nummern müssen selten auf die nächste Abfrage warten. Da Stationen mit niedrigen Nummern durchschnittlich $1{,}5N$ Zeitscheiben und solche mit hoher Nummer durchschnittlich $0{,}5N$ Zeitscheiben warten müssen, beträgt das Mittel für alle Stationen N Zeitscheiben.

Die Effizienz des Kanals bei niedriger Auslastung ist leicht zu berechnen. Der zusätzliche Platzbedarf pro Rahmen beträgt N Bit. Bei einer Datenmenge von d Bit ergibt sich eine Effizienz von $d/(N+d)$.

Bei hoher Belastung, wenn alle Stationen andauernd senden wollen, werden die N Bit der Konkurrenzperiode auf N Rahmen aufgeteilt. Dadurch erhöht sich der Platzbedarf nur um ein Bit pro Rahmen, was zu einer Effizienz von $d/(d+1)$ führt. Die durchschnittliche Übertragungsverzögerung eines Rahmens ist gleich der Zeit, die er innerhalb der Station in Warteposition steht, zuzüglich $(N-1)d+N$, wenn er einmal den Kopf der internen Warteschlange erreicht hat. Diese Zeitspanne gibt an, wie lange gewartet werden muss, bis alle anderen Stationen an der Reihe waren und einen Rahmen gesendet haben und ein weiteres Bitmuster durchgelaufen ist.

Tokenweitergabe

Das Wesentlich am Bitmusterprotokoll ist, dass es jeder Station der Reihe nach die Übertragung eines Rahmens in einer vorgegebenen Reihenfolge ermöglicht. Dasselbe kann auch auf einem anderen Weg erreicht werden, und zwar indem eine kleine Nachricht, ein **Token**, von einer Station zur nächsten in der gleichen Reihenfolge übergeben wird. Das Token repräsentiert die Erlaubnis zum Senden. Falls eine Station zu dem Zeitpunkt, wenn es das Token erhält, einen Rahmen in ihrer Warteschlange zur Übertragung bereit hat, dann kann die Station diesen Rahmen senden, bevor das Token zur nächsten Station weitergegeben wird. Falls kein Rahmen in der Warteschlange ist, wird das Token einfach weitergegeben.

Bei einem **Token-Ring**-Protokoll wird die Topologie des Netzwerks benutzt, um die Reihenfolge festzulegen, in der die Stationen senden. Die Stationen sind jeweils mit der nächsten zu einem Ring verbunden. Die Weitergabe des Token zur nächsten Station besteht dann einfach darin, das Token aus der einen Richtung zu empfangen und es in die andere Richtung weiterzusenden (▶Abbildung 4.7). Die Übermittlung der Rahmen findet ebenfalls in der Richtung des Token statt. Auf diese Art kreisen die Rahmen auf dem Ring herum und erreichen immer ihre Zielstation. Um zu verhindern, dass Rahmen (bzw. das Token) unendlich im Ring kreisen, muss irgendeine Station den Rahmen wieder vom Ring entfernen. Diese Station könnte entweder diejenige sein, die den Rahmen ursprünglich ausgesandt hat, nachdem er einen vollständigen Zyklus durchlaufen hat, oder die Station, die der beabsichtigte Empfänger des Rahmens war.

Copyright