Leaky-Bucket & Token-Bucket-Algorithmus

Ausarbeitung im Rahmen der Vorlesung

„Lokale Netze und Weitverkehrsnetze II“ (Prof. Koops)

SS 2001

Henning Jordan

Michael Solbach

Kay Süsens

Fachhochschule Oldenburg / Ostfriesland / **Wilhelmshaven**

Traffic-Shaping

Eines der Hauptprobleme in Netzen sind Verkehrsspitzen, also hohes Datenvolumen in kurzer Zeit. Dies kommt daher, dass der Datenverkehr nicht immer gleichmäßig fließt. Viele Überlastungen könnten vermieden werden, wenn die Hosts die Daten mit einer universellen rate übertragen würden. Der entsprechende Ansatz heißt Traffic-Shaping und wird hauptsächlich in ATM-Netzen angewandt.

Der Leaky-Bucket-Algorithmus

Der Leaky-Bucket-Algorithmus funktioniert im Prinzip wie ein Wassereimer mit einem kleinen Loch im Boden. Wenn nun in diesem Eimer Wasser vorhanden ist, läuft es mit einer gleichmäßigen Rate aus dem Loch aus, wenn kein Wasser im Eimer ist beträgt die Rate 0. Ist der Eimer voll, läuft zusätzlich Wasser über den Rand und geht verloren – kommt also nie am Loch an.

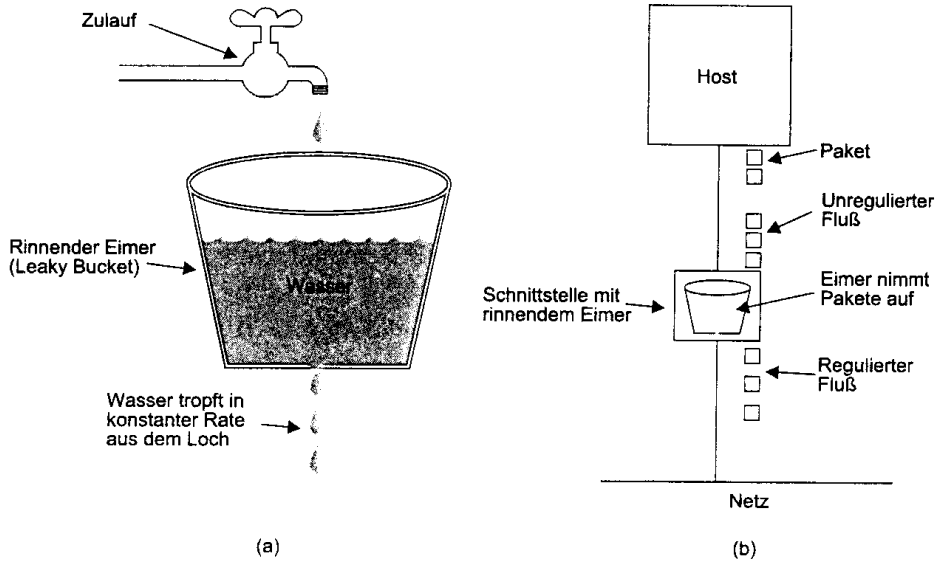

Abb. 1.- (a) Ein mit Wasser gefüllter rinnender Eimer (b) ein rinnender Eimer mit Paketen

Dieses Konzept können wir nun genauso gut auf Pakete in einem Netz anwenden. (Abb. 1b) Jeder Host ist im Prinzip über einen rinnenden Eimer ans Netz angeschlossen. Der Eimer stellt eine endliche interne Warteschleife dar, ist die Warteschleife belegt, werden zusätzlich ankommende Pakete verworfen. Dieses Verfahren kann entweder hardwareseitig implementiert werden oder vom Betriebssystem des Hosts simuliert werden. Vorgeschlagen wurde diese Methode erstmals von Turner (1986) und heißt Leaky-Bucket-Algorithmus. Es handelt sich dabei lediglich um ein System mit einer Warteschlange auf einem Server und einer konstanten Dienstzeit.

Pro Zeittakt kann der Host ein Paket in das Netz einspeisen. Dies wird, wie oben gesagt, durch die Hardware oder das Betriebssystemvorgegeben. Durch diesen Mechanismus wird stets eine gleichmäßiger Paketfluß vom Host in das Netz erzwungen.

Dadurch werden verkehrsspitzen geglättet und das Risiko von Überlastungen stark reduziert.
Bei Paketen stets gleicher Größe kann der Algorithmus wie gerade beschrieben eingesetzt werden. Bei Paketen variabler Größe ist es aber meist sinnvoller, eine feste Anzahl von Bytes pro Zeittakt zuzulassen. Dadurch können z.B. in einem Zeittakt, in dem 1024 Byte zugelassen sind, ein einzelnes 1024-Byte-Paket aber auch zwei 512-Byte-Pakete oder andere Kombinationen zugelassen werden. Stehen nicht mehr genügend freie Bytes zur Verfügung, muß das nächste Paket bis zum nächsten Zeittakt warten.

In seiner Ursprungsform ist der Leaky-Bucket-Algorithmus einfach zu implementieren. Der Eimer stellt eine endliche Warteschlange dar, in die ankommende Paketegelangen solange Platz vorhanden ist, ansonsten werden sie verworfen. Pro Zeittakt wird ein Paket aus der Warteschlange – sofern vorhanden – übertragen.

Der Leaky-Bucket-Algorithmus auf der Grundlage von Bytezahlen wird fast auf die gleiche Weise implementiert. Bei jedem Zeittakt wird ein Zähler auf n initialisiert. Umfaßt das erste Paket in der Warteschlange weniger Bytes als der momentan im Zähler stehende Wert, wird es übertragen, und der Zähler wird um diese Anzahl von Bytes erhöht. Solange der Zähler noch nicht seinen Schwellenwert erreicht hat, können weitere Pakete übertragen werden. Ist der Restwert im Zähler niedriger als die Länge des nächsten in der Warteschlange anstehenden Pakets, wird die Übertragung bis zum nächsten Zeittakt unterbrochen; dann wird die restliche Bytezahl überschrieben.
Als Beispiel eines rinnenden Eimers stelle man sich einen Rechner vor, der Daten in 25 Millionen Byte/s (200 Mbps) produziert, und ein Netz, das in der gleichen Geschwindigkeit läuft. Die Router können diese Datenrate nur über kurze Intervalle unterstützen. Bei längeren Intervallen funktionieren sie am besten bei Raten, die 2 Millionen Byte/s nicht übersteigen. Nun nehmen wir an, daß Daten in Spitzen von je einer Million Byte, d.h. eine 40-msSpitze pro Sekunde, einfließen. Um die Durchschnittsrate auf 2 Mbyte/s zu reduzieren, könnte man einen rinnenden Eimer mit o = 2 Mbyte/s und eine Kapazität C von 1 Mbyte zugrunde legen. Das bedeutet, daß Spitzen von bis zu 1 Mbyte ohne Datenveriust bewältigt werden und daß sich solche Spitzen über 500 ins verteilen, gleichgültig, wie schnell sie ankommen.
In Abb. 5.25(c) sehen wir den Eingan- in den rinnenden Eimer mit 25 Mbyte/s über 40 ins
In Abb. 5.25(b) ist der Ausgang in einer gleichmäßigen Rate von 2 Mbyte/s über 500 ins dargestellt.

5.3.3.2 Der Token-Bucket-Algorithmus

Der Leaky-Bucket-Algorithmus zwingt ein straffes Ausgabemuster mit einer Durchschnittsrate auf, gleichgültig, welche Verkehrsspitzen auftreten. Bei vielen Anwendungen ist es besser, die Beschleunigung von Ausgaben bei bestimmten Verkehrsspitzen zuzulassen. Dafür wird ein fle:~iblerer Algorithmus benötigt, vorzugsweise ei-

ner, der keine Daten verliert. Ein solcher Algorithmus ist der Token-Bucket-Algorithmus. Bei diesem Algorithmus hält der rinnende Eimer Token, die von einer Uhr mit einer Geschwindigkeit von einem Token alle JT Sekunden erzeugt werden. In Abb. 5.26(a) sehen wir einen Eimer mit drei Token, während fünf Pakete auf ihre Übertragung warten. Damit ein Paket übertragen werden kann, muß es ein Token erfassen und zerstören. In Abb. 5.26(b) sind drei der fünf Pakete durchgekommen, aber die anderen zwei warten noch, bis weitere Token erzeugt werden.
Der Token-Bucket-Algorithmus bietet eine andere Form der Verkehrsgestaltung als der Leaky-Bucket-Algorithmus. Letzterer läßt nicht zu, daß Hosts träge bleiben, Pakete ansammeln und diese dann mit hohen Verkehrsspitzen alle auf einmal übertragen. Der TokenBucket-Algorithmus läßt diese Ansammeln zu, aber nur bis zur maximalen Eimergröße von n. Dieses Merkmal bedeutet, daß Spitzen von bis zu n Paketen gleichzeitig übertragen werden können, so daß auch auf dem Ausgabestrom gewisse Spitzen vorkommen, was aber zu schnelleren Bestätigungen führt.

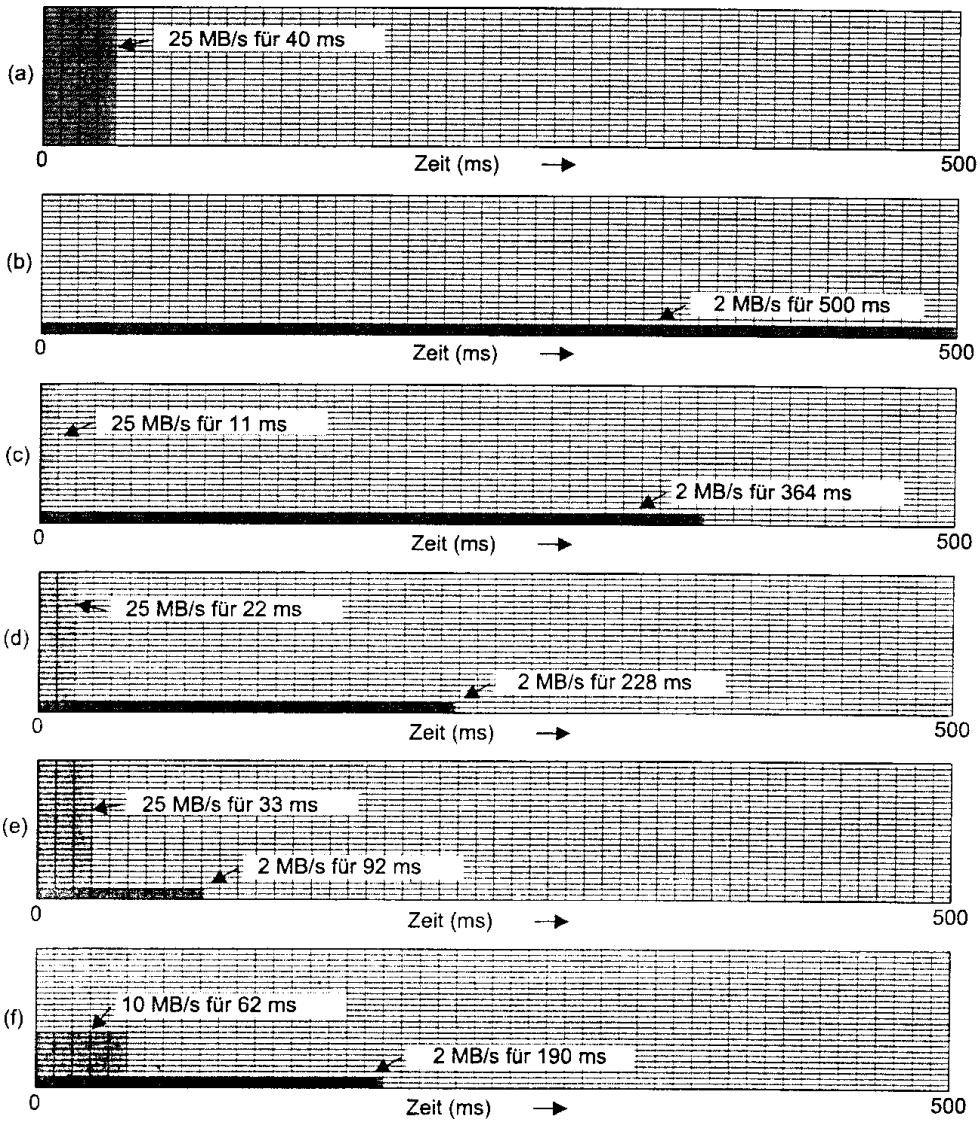

Abb. 5.25: (a) Zufluß in den rinnenden Eimer; (b) Abfluß vom rinnenden Eimer; (c) - (e) Ausgabe von einem Token-Bucket mit Kapazitäten von 250, 500 und 750 Kbyte; (f) Ausgabe von einem Token-Bucket mit 500 KbYte und Speisung eines rinnenden Eimers mit 10 Mbytels

Ein weiterer Unterschied zwischen den zwei Algorithmen ist der, daß der Token-BucketAlgorithmus Token wegwirft, wenn der Eimer voll ist, aber nie Pakete. Demgegenüber werden beim Leaky-Bucket-Algorithmus Pakete weggeworfen, wenn der Eimer voll ist.

Auch hier ist eine geringfügige Variante möglich, bei der jedes Token dazu berechtigt, anstelle eines Pakets k Byte zu übertragen. Ein Paket kann nur übertragen werden, wenn genügend Token verfügbar sind, um die Länge in Bytes abzudecken. Angebrochene Token werden zur späteren Verwendung aufgehoben.

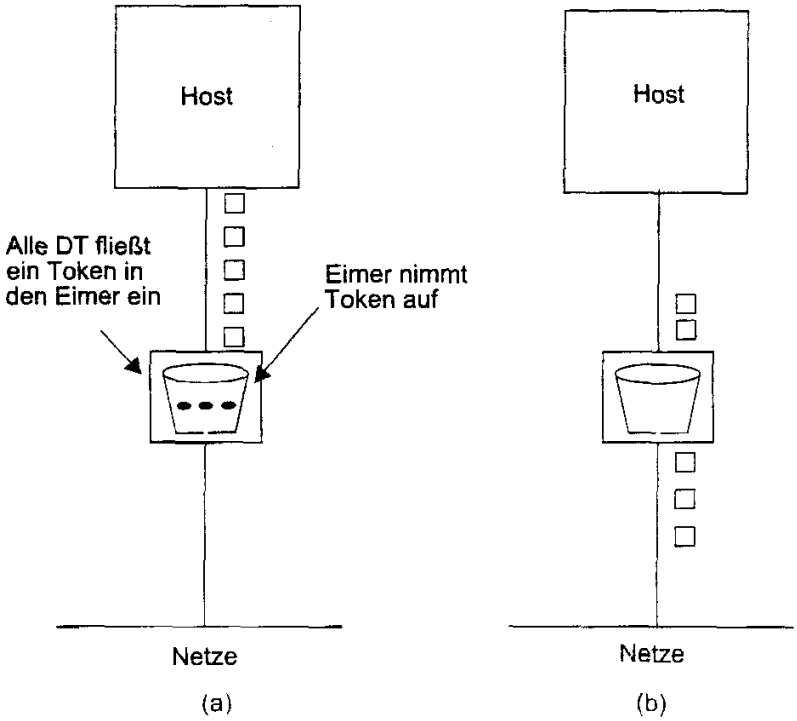

Abb. 5."6: Der Token -Bucket-A lgo rithm its,- (a) vorher,- (b) nachher

Beide Bucket-Algorithmen eignen sich auch zum Ausgleichen von Verkehrsspitzen zwischen Routern und natürlich zum Regulieren der Hostausgaben, wie in unserem Beispiel.

Ein deutlicher Unterschied liegt darin, daß ein Token-Bucket, der einen Host reguliert, den Host je nach Regel vom weiteren Ubertragen abhalten kann. Wird ein Router auf diese Weise daran gehindert, weiterzusenden, können Daten verlorengehen.

Die Implementierung des Token-Bucket-Algorithmus in seiner Grundform besteht lediglich in einer Variablen für die Zählung der Token. Der Zähler wird einmal alle AT erhöht und nach der Versendung eines Pakets je um eins gesenkt. Fällt der Zähler auf Null, dürfen keine Pakete mehr übertragen werden. Bei der Variante mit Bytezählung wird der Zähler alle ‚AT um k Byte erhöht und um die Länge jedes gesendeten Pakets gesenkt.

Im Grunde läßt der Token-Bucket Verkehrsspitzen zu, aber nur bis zu einer bestimmten Höchstlänge. In Abb. 5.25(c) ist z.B. ein Token-Bucket mit einer Kapazität von 250 Kbyte dargestellt. Die Token kommen mit einer Rate ein, die eine Ausgabe von 2 Mbyte/s zuläßt. Unter der Annahme, daß der Token-Bucket voll ist, wenn 1 Mbyte ankommt, kann der Eimer die vollen 25 Mbyte/s etwa 11 ms lang ablassen. Dann muß er auf 2 Mbyte/s zurückdrehen, bis das gesamte Eingangsvolumen übertragen ist.

Der Token-Bucket-Algorithmus ist dahingehend potentiell problematisch, daß er wieder große Spitzen zuläßt, obwohl das Spitzenintervall durch sorgfältige Auswahl von o und M Cr reguliert werden kann. Häufig ist es wünschenswert, die Spitzenrate

zu reduzieren, aber nicht zurück bis zum niedrigsten Wert des ursprünglichen Leaky-Bucket.
Eine Möglichkeit, den Verkehr auszugleichen, besteht durch Einfügen eines rinnenden Eimers nach dem Token-Bucket. Die Rate des rinnenden Eimers sollte höher sein als 0 des Token-Buckets, aber niedriger als die Höchstrate des Netzes. Abb. 5.'-25(f) zeigt den Ausgang für einen Token-Bucket mit 500 Kbyte, gefolgt von einem rinnenden Eimer mit XXX 10 Mbyte/s.
Die Regulierung dieser Schemata ist nicht immer einfach. Im wesentlichen muß das Netz den Algorithmus simulieren und sicherstellen, daß nicht mehr Pakete oder Bytes als zulässig übertragen werden. Überschüssige Pakete werden verworfen oder beschädigt, wie später noch aufgezeigt wird.

Flußspezifikationen

Die Traffic-Shaping-Methode ist am wirksamsten, wenn Sender, Empfänger und Teilnetz aeineinsam eine Vereinbarung treffen. Um eine Übereinkunft zu erreichen, muß das Verkehrsmuster ganz genau definiert werden. Eine solche Übereinkunft nennt man Flußspezifikation (Flow Specification). Sie besteht aus einer Datenstruktur, die sowohl das Muster des eingespeisten Verkehrs als auch die von den Anwendungen gewünschte Dienstqualität beschreibt. Eine Flußspezifikation kann auf die über eine virtuelle Verbindung gesendeten Pakete oder auf eine Folge von Datengrammen von einer Quelle an ein oder mehrere Ziele zutreffen.