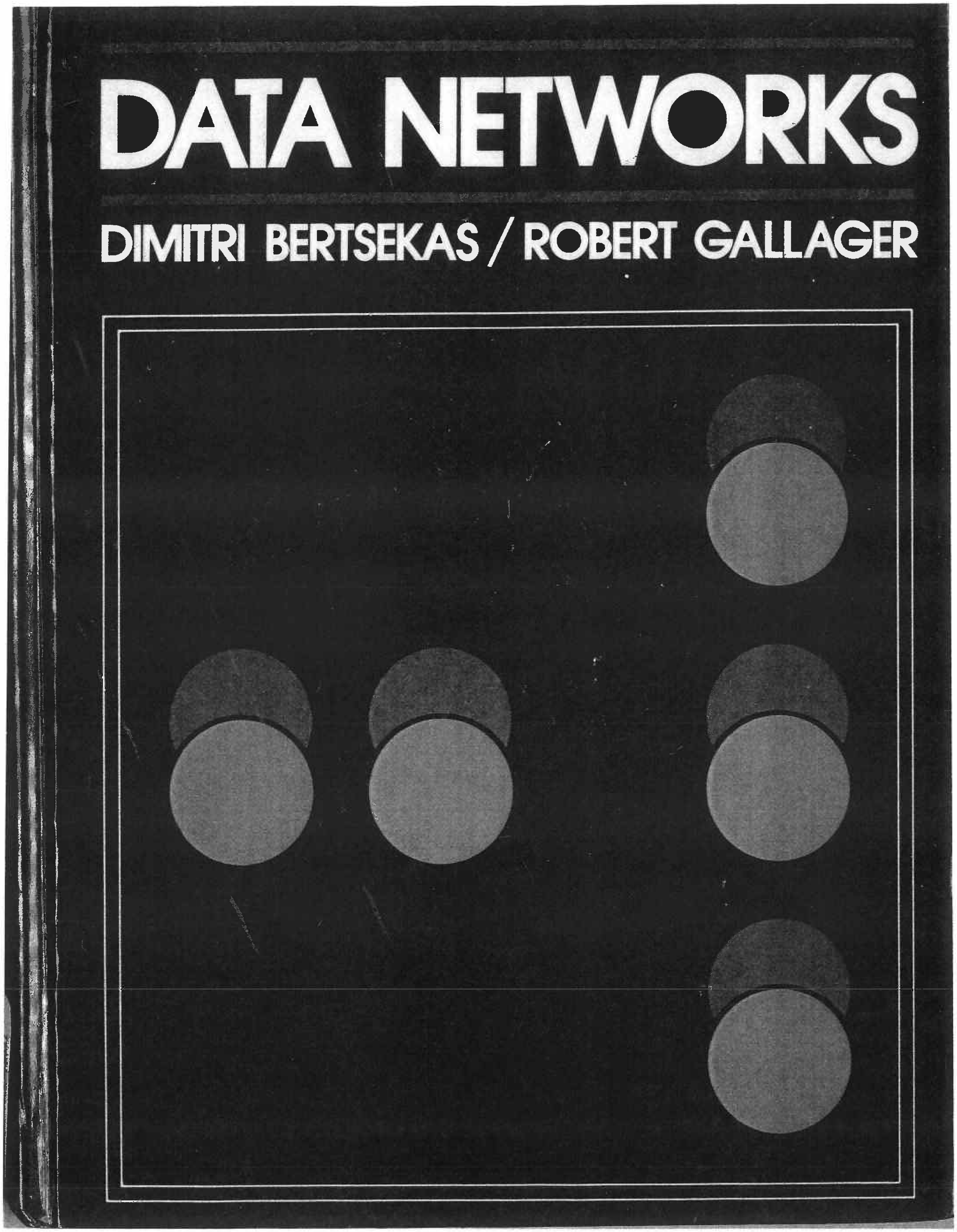

DATA NETWORKS

DIMITRI BERTSEKAS / ROBERT GALLAGER

BROADCOM 1012

To Joanna and Marie

Editorial/production supervision by Margaret Rizzi
Cover design by Wanda Lubelska
Manufacturing buyer: Rhett Conklin

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-196825-4 025

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, London
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, Sydney
PRENTICE-HALL CANADA INC., Toronto
PRENTICE-HALL HISPANOAMERICANA, S.A., Mexico
PRENTICE-HALL OF INDIA PRIVATE LIMITED, New Delhi
PRENTICE-HALL OF JAPAN, INC., Tokyo
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., Singapore
EDITORA PRENTICE-HALL DO BRASIL, LTD., Rio de Janeiro

the above properties). Their generator polynomials are

$$g(D) = D^{16} + D^{15} + D^2 + 1, \quad \text{for CRC-16}$$
$$g(D) = D^{16} + D^{12} + D^5 + 1, \quad \text{for CRC-CCITT}$$

2.4 ARQ—RETRANSMISSION STRATEGIES

The general concept of *a*utomatic *r*epeat re*q*uest (ARQ) is to detect frames with errors at the receiving DLC module and then to request the transmitting DLC module to repeat the information in those erroneous frames. Error detection was discussed in the last section, and the problem of requesting retransmissions is treated in this section. There are two quite different aspects of retransmission algorithms or protocols. The first is that of correctness; *i.e.*, does the protocol succeed in releasing each packet, once and only once, without errors, from the receiving DLC? The second is that of efficiency; *i.e.*, how much of the bit transmitting capability of the bit pipe is wasted by unnecessary waiting and by sending unnecessary retransmissions? First, several classes of protocols are developed and shown to be correct (in a sense to be defined more precisely later). Later, the effect that the various parameters in these classes have on efficiency is considered.

Recall from Fig. 2.2 that packets enter the DLC layer from the network layer; a header and trailer are appended to each packet in the DLC module to form a frame and the frames are transmitted on the virtual bit pipe (*i.e.*, are sent to the physical layer for transmission). When errors are detected in a frame, a new frame containing the old packet is transmitted. Thus, the first transmitted frame might contain the first packet, the next frame the second packet, the third frame a repetition of the first packet, and so forth. When a packet is repeated, the header and trailer might or might not be the same as in the earlier version.

Since framing will not be discussed until the next section, we continue to assume that the receiving DLC knows when frames start and end; thus a CRC (or any other technique) may be used for detecting errors. We also assume, somewhat unrealistically, that *all* frames with errors are detected. The reason for this is that we want to prove that ARQ works correctly except when errors are undetected. This is the best that can be hoped for, since error detection cannot work with perfect reliability and bounded delay; in particular, any code word can be changed into another code word by some string of transmission errors. This can cause erroneous data to leave the DLC or, perhaps worse, can cause some control bits to be changed.

Next, a number of assumptions are made about the bit pipes over which these frames are traveling. These assumptions are somewhat more general than before. The major reason for this generality will appear when framing is studied; in effect, these general conditions will allow us to relax the assumption that the receiving DLC has framing information.

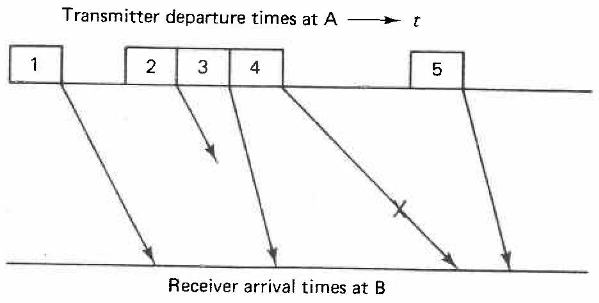
Transmitter departure times at A ⟶ t
1
2
3
4
5
Receiver arrival times at B

Figure 2.17 Model of frame transmissions: frame 2 is lost and never arrives; frame 4 contains errors; the frames have variable transmission delay, but those that arrive do so in the order transmitted.

Assume first that each transmitted frame is delayed by an arbitrary and variable time before arriving at the receiver, and assume that some frames might be "lost" and never arrive. Those frames that arrive, however, are assumed to arrive in the same order as transmitted, with or without errors. Figure 2.17 illustrates this behavior.

2.4.1 *Stop-and-Wait ARQ*

The simplest type of retransmission protocol is called *stop-and-wait*. The basic idea is to ensure that each packet has been correctly received before initiating transmission of the next packet. Thus, in transmitting packets from point A to B, the first packet is transmitted in the first frame, and then the sending DLC waits. If the frame is correctly received at B, B sends an acknowledgment (called an ack) back to A; if the frame is received with errors, B sends a negative acknowledgment (called a nak) back to A. Since errors can occur from B to A as well as from A to B, the ack or nak is protected with a CRC.

If a frame is received without errors at B, and the corresponding ack is received without errors at A, then A can start to send the next packet in a new frame. Alternatively, detected errors can occur either in the transmission of the frame or the return ack or nak, and in this case A resends the old packet in a new frame. Finally, if either the frame or the return ack or nak is lost, A must eventually time out and resend the old packet. Since frames can be arbitrarily delayed, it is possible for A to time out, to resend an old packet, and subsequently to get an ack or nak for the earlier transmission (see Fig. 2.18).

Note that because of the time outs, and also because of the potential errors in the B to A channel, it is possible that B receives the first packet correctly in the first frame and then receives it again in the second frame (see Fig. 2.18). By assumption, packets are arbitrary bit strings, and the first and second packet could be identical. Thus, if B receives the same packet twice, it cannot tell whether

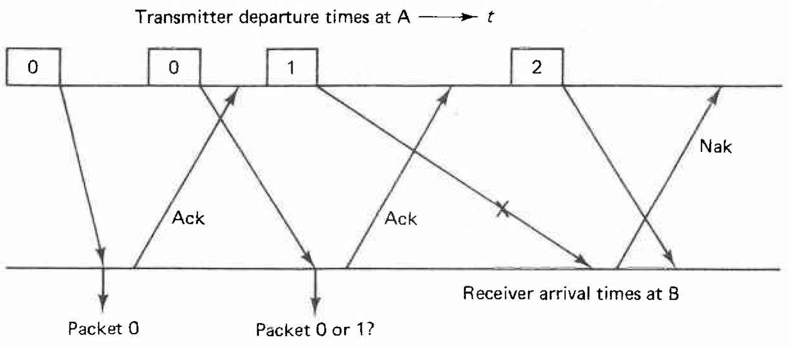
Transmitter departure times at A → t
0
0
1
2
Ack
Ack
Nak
Receiver arrival times at B
Packet 0
Packet 0 or 1?

Figure 2.18 The trouble with unnumbered packets. If the transmitter at *A* times out and sends packet 0 twice, then the receiver at *B* cannot tell whether the second frame is a retransmission of packet 0 or the first transmission of packet 1.

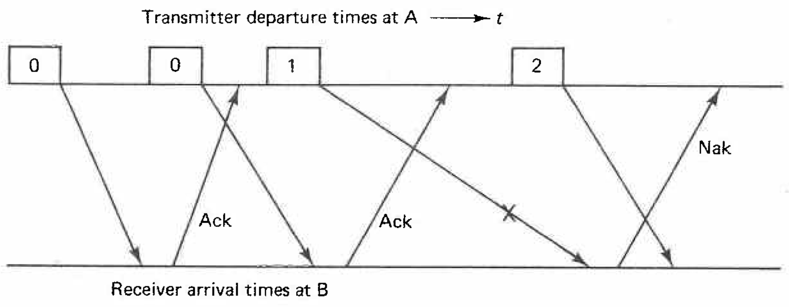
Transmitter departure times at A → t
0
0
1
2
Ack
Ack
Nak
Receiver arrival times at B

Figure 2.19 The trouble with unnumbered acks. If the transmitter at *A* times out and sends packet 0 twice, and then sends packet 1 after the first ack, node *A* cannot tell whether the second ack is for packet 0 or 1.

the second frame is repeating the first packet or sending the second packet (which happens to be the same bit string as the first packet).

The simplest solution to this problem is for the sending DLC module (at *A*) to add a sequence number to identify successive packets. Although this strategy seems simple and straightforward, it will not work as just described. The problem is that acks can get lost on the return channel, and thus when *B* gets the same packet correctly twice in a row, it has to send a new ack for the second reception (see Fig. 2.19). Node *A*, after transmitting the packet twice but receiving only one ack, could transmit the next packet in sequence, and then on receiving the second ack, could interpret that as an ack for the new packet, leading to a potential failure of the system.

To avoid this type of problem, the receiving DLC (at *B*), instead of returning ack or nak on the reverse link, returns the number of the next packet awaited. This

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE