Chapter Six

Errors, Error Detection, and Error Control

Data Communications and Computer Networks: A Business User's Approach

Eighth Edition

After reading this chapter, you should be able to:

- Identify the different types of noise commonly found in computer networks
- Specify the different error-prevention techniques, and be able to apply an error-prevention technique to a type of noise
- Compare the different error-detection techniques in terms of efficiency and efficacy
- Perform simple parity and longitudinal parity calculations, and enumerate their strengths and weaknesses

After reading this chapter, you should be able to (continued):

- Cite the advantages of arithmetic checksum
- Cite the advantages of cyclic redundancy checksum, and specify what types of errors cyclic redundancy checksum will detect
- Differentiate between the basic forms of error control, and describe the circumstances under which each may be used
- Follow an example of a Hamming self-correcting code

Introduction

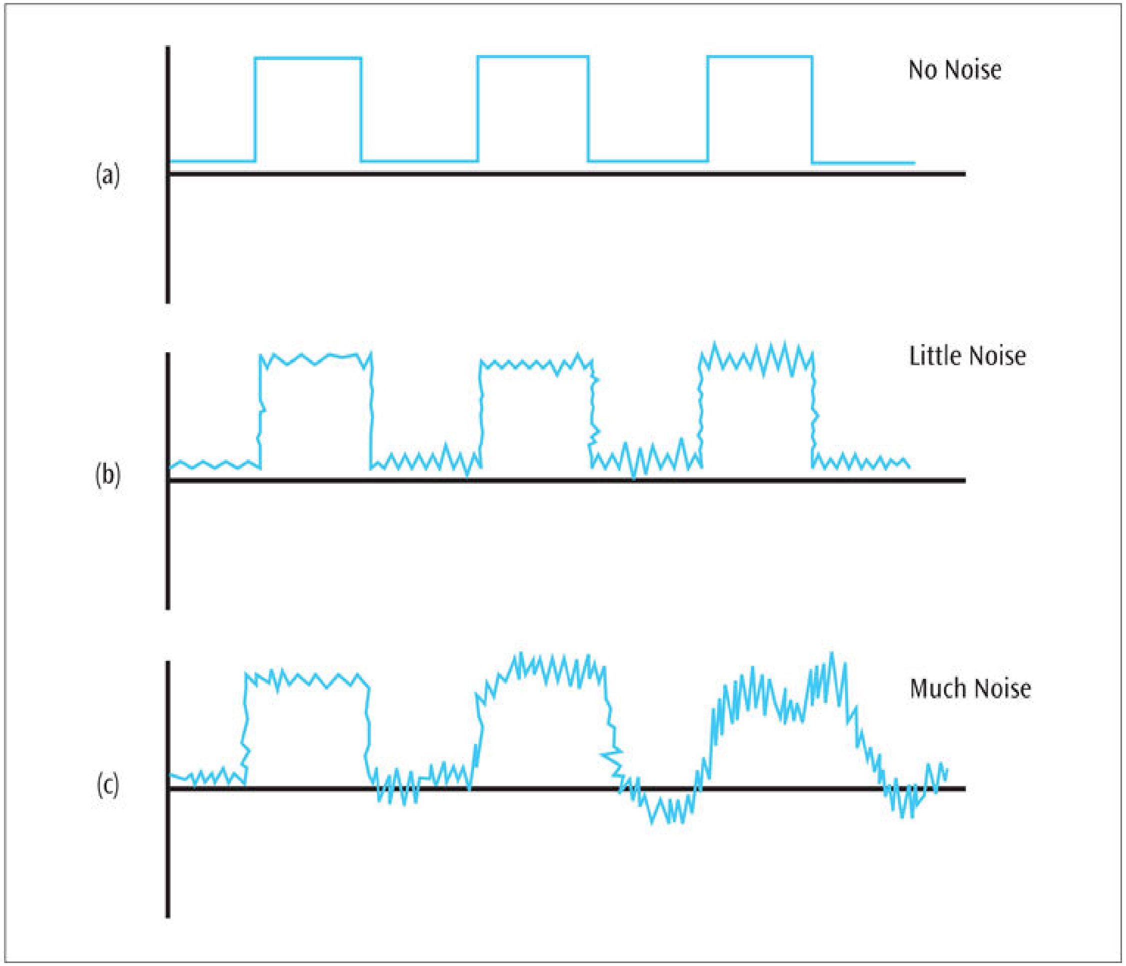
- Noise is always present
- If a communications line experiences too much noise, the signal will be lost or corrupted
- Communication systems should check for transmission errors
- Once an error is detected, a system may perform some action
- Some systems perform no error control, but simply let the data in error be discarded

White Noise

- Also known as thermal or Gaussian noise
- Relatively constant and can be reduced
- If white noise gets too strong, it can completely disrupt the signal

White Noise (continued)

Figure 6-1
White noise as it interferes with a digital signal

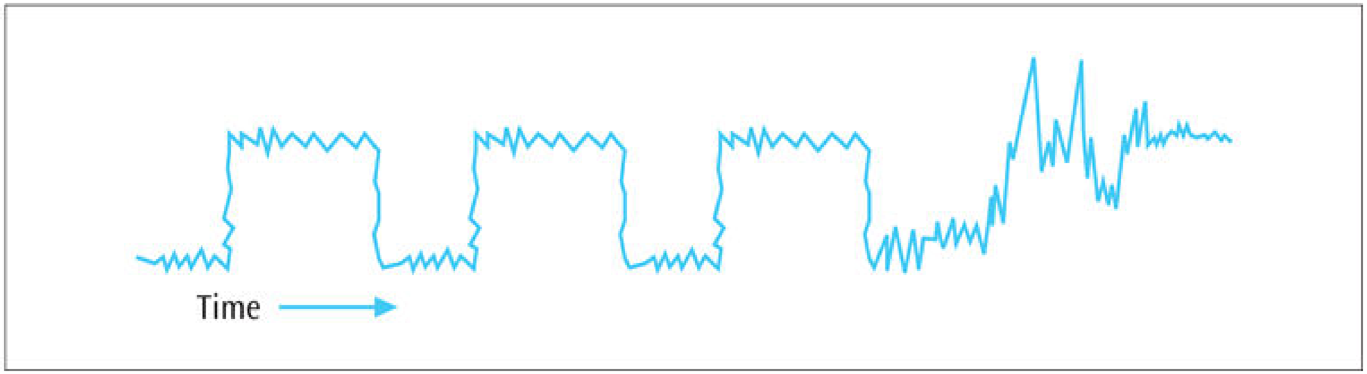


Impulse Noise

- One of the most disruptive forms of noise
- Random spikes of power that can destroy one or more bits of information
- Difficult to remove from an analog signal because it may be hard to distinguish from the original signal
- Impulse noise can damage more bits if the bits are closer together (transmitted at a faster rate)

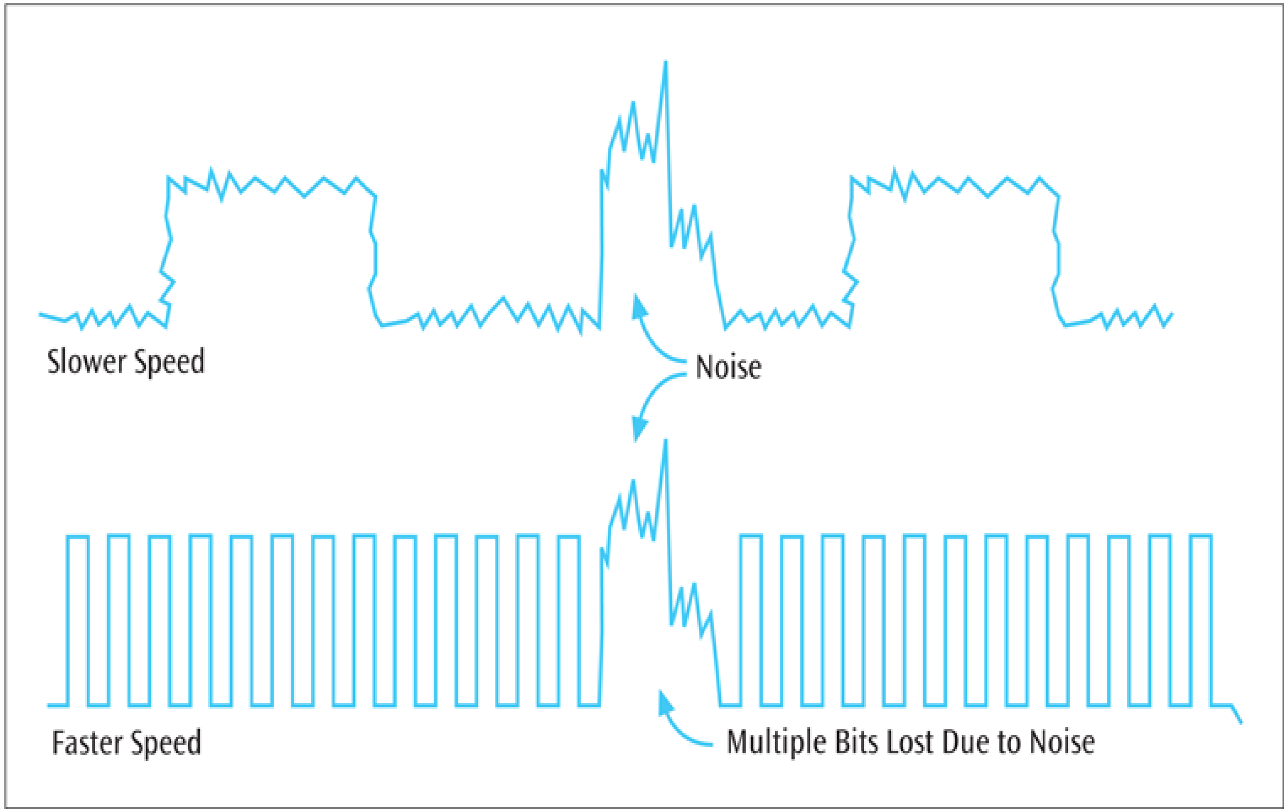
Impulse Noise (continued)

Figure 6-2
The effect of impulse noise on a digital signal



Impulse Noise (continued)

Figure 6-3
Transmission speed and its relationship to noise in a digital signal

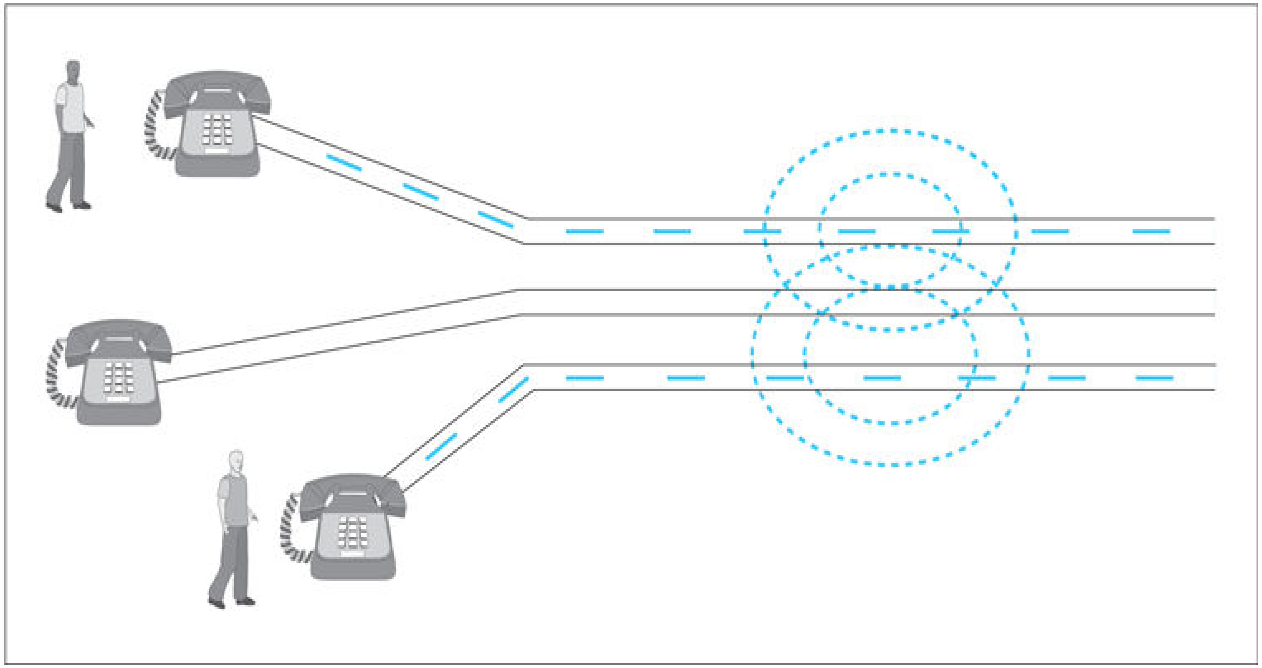


Crosstalk

- Unwanted coupling between two different signal paths
  - For example, hearing another conversation while talking on the telephone
- Relatively constant and can be reduced with proper measures

Crosstalk (continued)

Figure 6-4
Three telephone circuits experiencing crosstalk

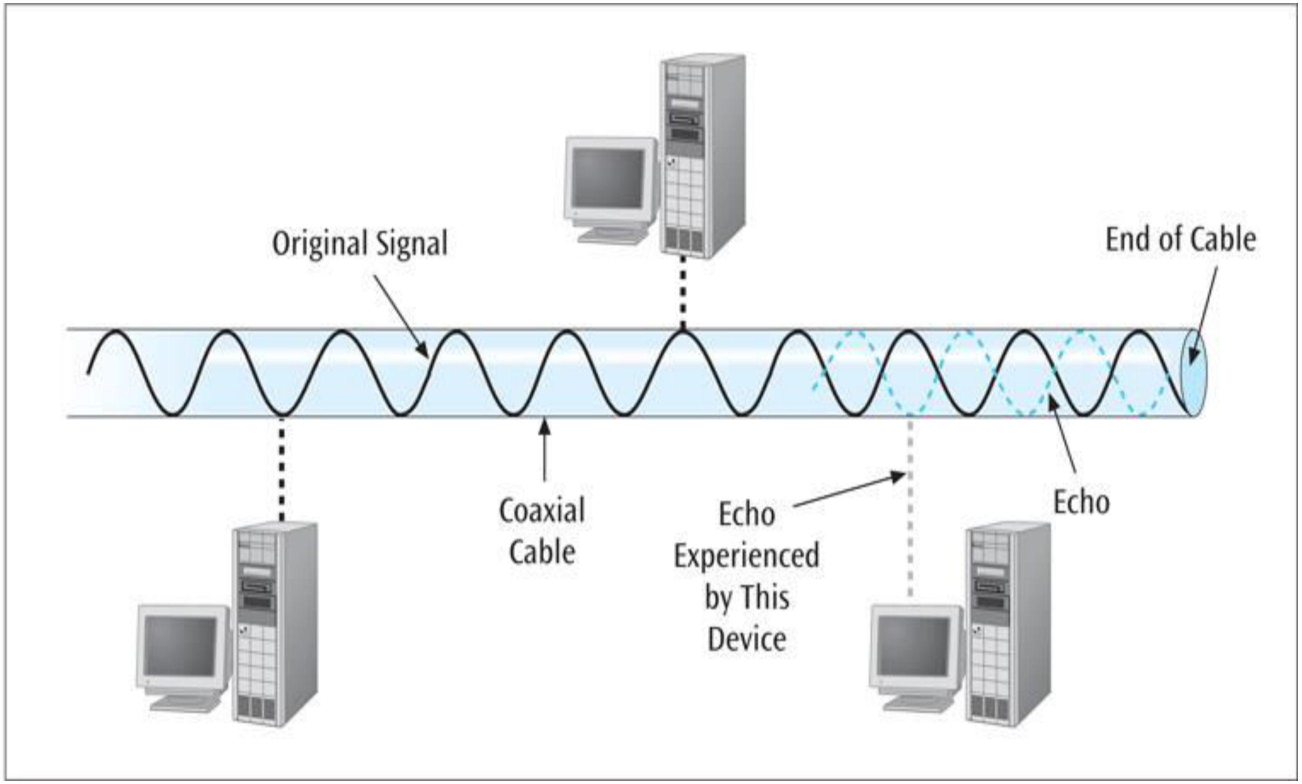


Echo

- The reflective feedback of a transmitted signal as the signal moves through a medium
- Most often occurs on coaxial cable
- If echo bad enough, it could interfere with original signal
- Relatively constant, and can be significantly reduced

Echo (continued)

Figure 6-5
A signal bouncing back at the end of a cable and causing echo

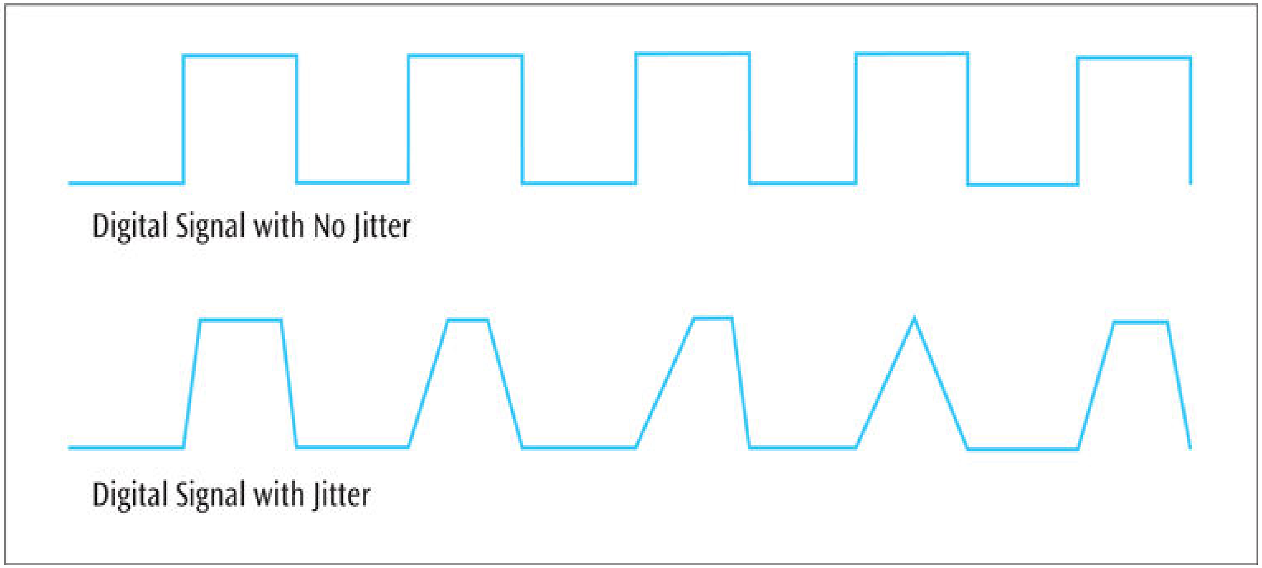


Jitter

- The result of small timing irregularities during the transmission of digital signals
- Occurs when a digital signal is repeated over and over
- If serious enough, jitter forces systems to slow down their transmission
- Steps can be taken to reduce jitter

Jitter (continued)

Figure 6-6
Original digital signal and digital signal with jitter



Delay Distortion

- Occurs because the velocity of propagation of a signal through a medium varies with the frequency of the signal
  - Can be reduced

Attenuation

- The continuous loss of a signal's strength as it travels through a medium

Error Prevention

- To prevent errors from happening, several techniques may be applied:
  - Proper shielding of cables to reduce interference
  - Telephone line conditioning or equalization
  - Replacing older media and equipment with new, possibly digital components
  - Proper use of digital repeaters and analog amplifiers
  - Observe the stated capacities of the media

Error Prevention (continued)

Table 6-1

Summary of errors and error-prevention techniques

Type of Error	Error-Prevention Technique
White noise	Install special filters for analog signals; implement digital signal regeneration for digital signals
Impulse noise	Install special filters for analog signals; implement digital signal processing for digital signals
Crosstalk	Install proper shielding on cables
Echo	Install proper termination of cables
Jitter	Use better-quality electronic circuitry, use fewer repeaters, slow the transmission speed
Attenuation*	Install device that amplifies analog signals; implement digital signal regeneration of digital signals

* Not a type of error, but indirectly affects error

Error Detection

- Despite the best prevention techniques, errors may still happen
- To detect an error, something extra has to be added to the data/signal
  - This extra is an error detection code
- Three basic techniques for detecting errors: parity checking, arithmetic checksum, and cyclic redundancy checksum

Parity Checks

- Simple parity
  - If performing even parity, add a parity bit such that an even number of 1s are maintained
  - If performing odd parity, add a parity bit such that an odd number of 1s are maintained
  - For example, send 1001010 using even parity
  - For example, send 1001011 using even parity

Parity Checks (continued)

- Simple parity (continued)
  - What happens if the character 10010101 is sent and the first two 0s accidentally become two 1s?
    - Thus, the following character is received: 11110101
    - Will there be a parity error?
      - Problem: Simple parity only detects odd numbers of bits in error

Parity Checks (continued)

- Longitudinal parity
  - Adds a parity bit to each character then adds a row of parity bits after a block of characters
  - The row of parity bits is actually a parity bit for each “column” of characters
  - The row of parity bits plus the column parity bits add a great amount of redundancy to a block of characters

Parity Checks (continued)

Table 6-2

Simple example of longitudinal parity

	Data							Parity
Row 1	1	1	0	1	0	1	1	1
Row 2	1	1	1	1	1	1	1	1
Row 3	0	1	0	1	0	1	0	1
Row 4	0	0	1	1	0	0	1	1
Parity Row	0	1	0	0	1	1	1	0

Parity Checks (continued)

Table 6-3

The second and third bits in Rows 1 and 2 have errors, but longitudinal parity does not detect the errors

	Data							Parity
Row 1	1	~~1~~ 0	~~0~~ 1	1	0	1	1	1
Row 2	1	~~1~~ 0	~~1~~ 0	1	1	1	1	1
Row 3	0	1	0	1	0	1	0	1
Row 4	0	0	1	1	0	0	1	1
Parity Row	0	1	0	0	1	1	1	0

Parity Checks (continued)

- Both simple parity and longitudinal parity do not catch all errors
- Simple parity only catches odd numbers of bit errors
- Longitudinal parity is better at catching errors but requires too many check bits added to a block of data
- We need a better error detection method
  - What about arithmetic checksum?

Arithmetic Checksum

- Used in TCP and IP on the Internet
- Characters to be transmitted are converted to numeric form and summed
- Sum is placed in some form at the end of the transmission

Arithmetic Checksum

- Simplified example:

$$\begin{array}{r} 56 \\ 72 \\ 34 \\ \underline{48} \\ 210 \end{array}$$

Then bring 2 down and add to right-most position

$$\begin{array}{r} 10 \\ \underline{\quad 2} \\ 12 \end{array}$$

Arithmetic Checksum

- Receiver performs same conversion and summing and compares new sum with sent sum
- TCP and IP processes a little more complex but idea is the same
- But even arithmetic checksum can let errors slip through. Is there something more powerful yet?

Cyclic Redundancy Checksum

- CRC error detection method treats the packet of data to be transmitted as a large polynomial
- Transmitter takes the message polynomial and using polynomial arithmetic, divides it by a given generating polynomial
- Quotient is discarded but the remainder is "attached" to the end of the message

Cyclic Redundancy Checksum (continued)

- The message (with the remainder) is transmitted to the receiver
- The receiver divides the message and remainder by the same generating polynomial
- If a remainder not equal to zero results, there was an error during transmission
- If a remainder of zero results, there was no error during transmission

Cyclic Redundancy Checksum (continued)

- Some standard generating polynomials:
- CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x + 1$
- CRC-16: $x^{16} + x^{15} + x^2 + 1$
- CRC-CCITT: $x^{16} + x^{15} + x^5 + 1$
- CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- ATM CRC: $x^8 + x^2 + x + 1$

Cyclic Redundancy Checksum (continued)

Table 6-4

Error-detection performance of cyclic redundancy checksum

Type of Error	Error Detection Performance
Single-bit errors	100 percent
Double-bit errors	100 percent, as long as the generating polynomial has at least three 1s (they all do)
Odd number of bits in error	100 percent, as long as the generating polynomial contains a factor $x + 1$ (they all do)
An error burst of length $< r + 1$	100 percent
An error burst of length $= r + 1$	probability $= 1 - (1/2)^{(r-1)}$ (very near 100%)
An error burst of length $> r + 1$	probability $= 1 - (1/2)^{r}$ (very near 100%)

Error Control

- Once an error is detected, what is the receiver going to do?
  - Do nothing (simply toss the frame or packet)
  - Return an error message to the transmitter
  - Fix the error with no further help from the transmitter

Do Nothing (Toss the Frame/Packet)

- Seems like a strange way to control errors but some lower-layer protocols such as frame relay perform this type of error control
- For example, if frame relay detects an error, it simply tosses the frame
  - No message is returned
- Frame relay assumes a higher protocol (such as TCP/IP) will detect the tossed frame and ask for retransmission

Return A Message

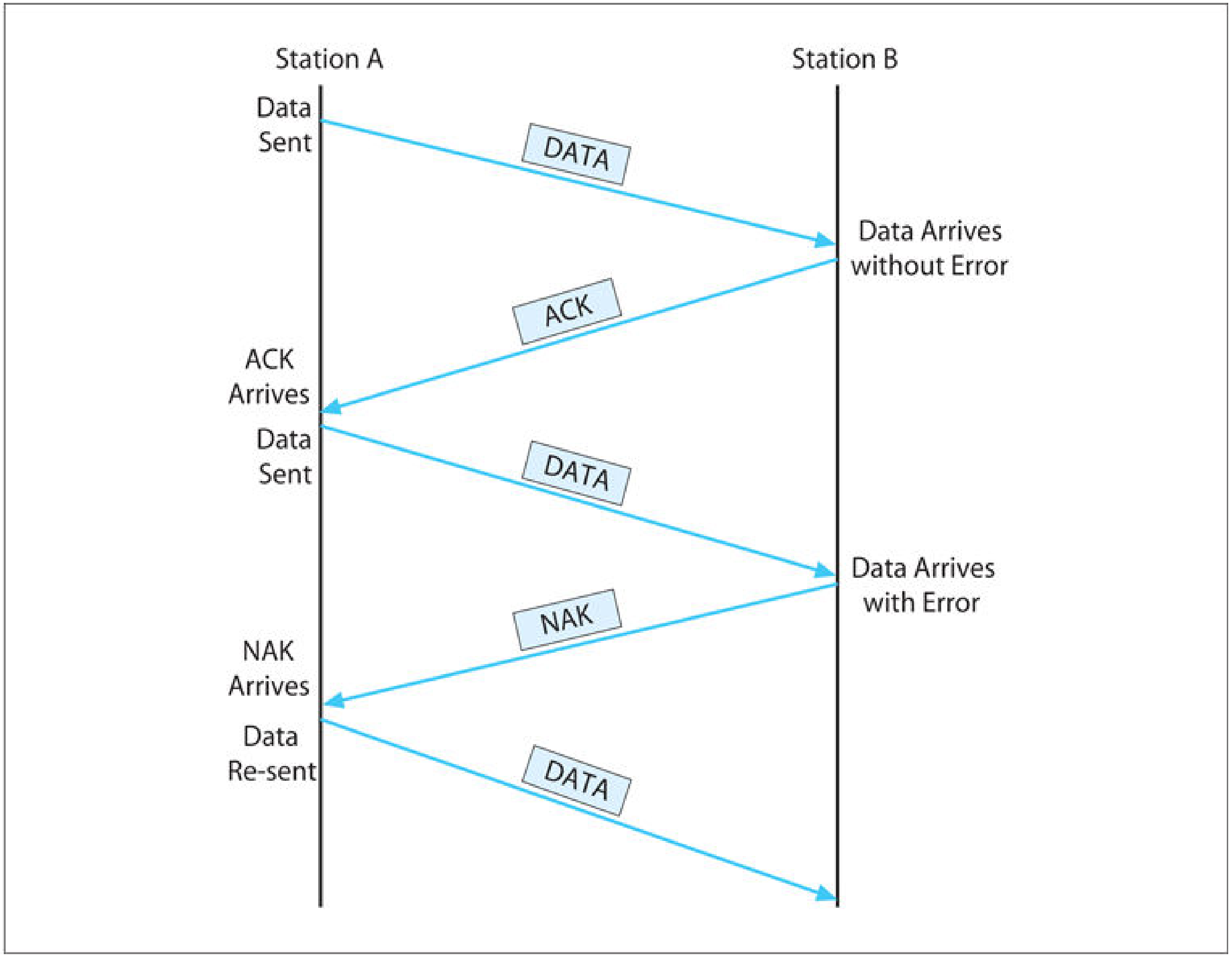
- Once an error is detected, an error message is returned to the transmitter
- Two basic forms:
  - Stop-and-wait error control
  - Sliding window error control

Stop-and-Wait Error Control

- Stop-and-wait is the simplest of the error control protocols
- A transmitter sends a frame then stops and waits for an acknowledgment
  - If a positive acknowledgment (ACK) is received, the next frame is sent
  - If a negative acknowledgment (NAK) is received, the same frame is transmitted again

Stop-and-Wait Error Control (continued)

Figure 6-8
Sample dialog using Stop-and-wait error control

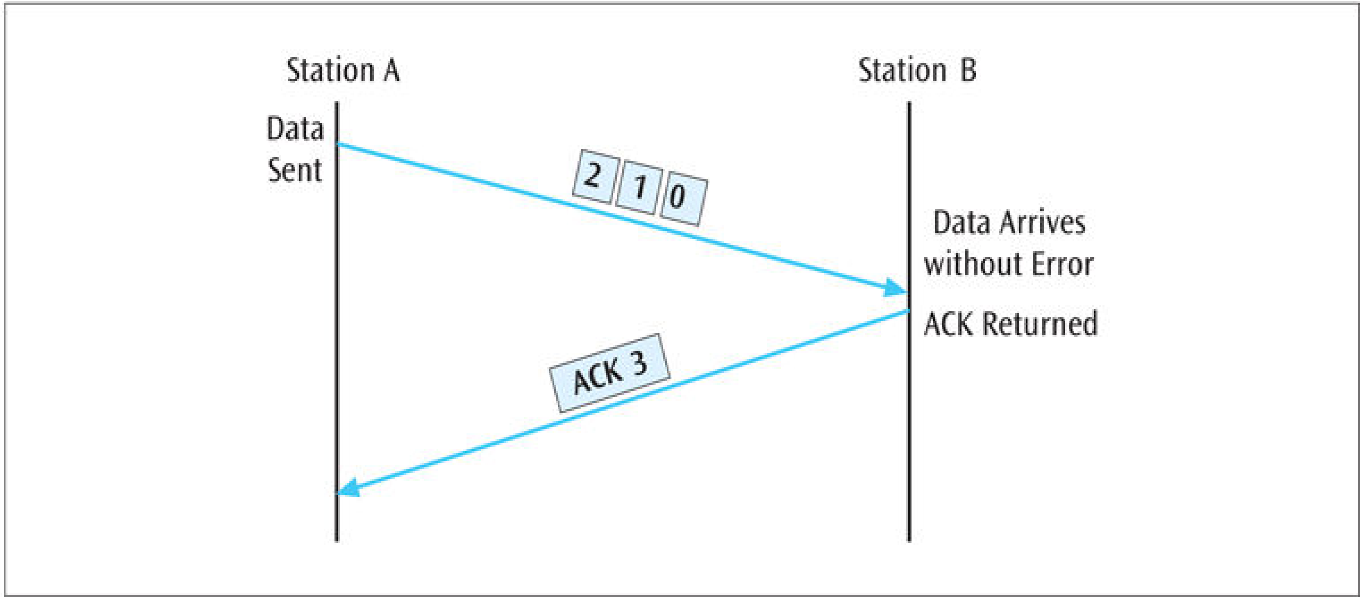


Sliding Window Error Control

- These techniques assume that multiple frames are in transmission at one time
- A sliding window protocol allows the transmitter to send a number of data packets at one time before receiving any acknowledgments
  - Depends on window size
- When a receiver does acknowledge receipt, the returned ACK contains the number of the frame expected next

Sliding Window Error Control (continued)

Figure 6-9
Example of sliding window

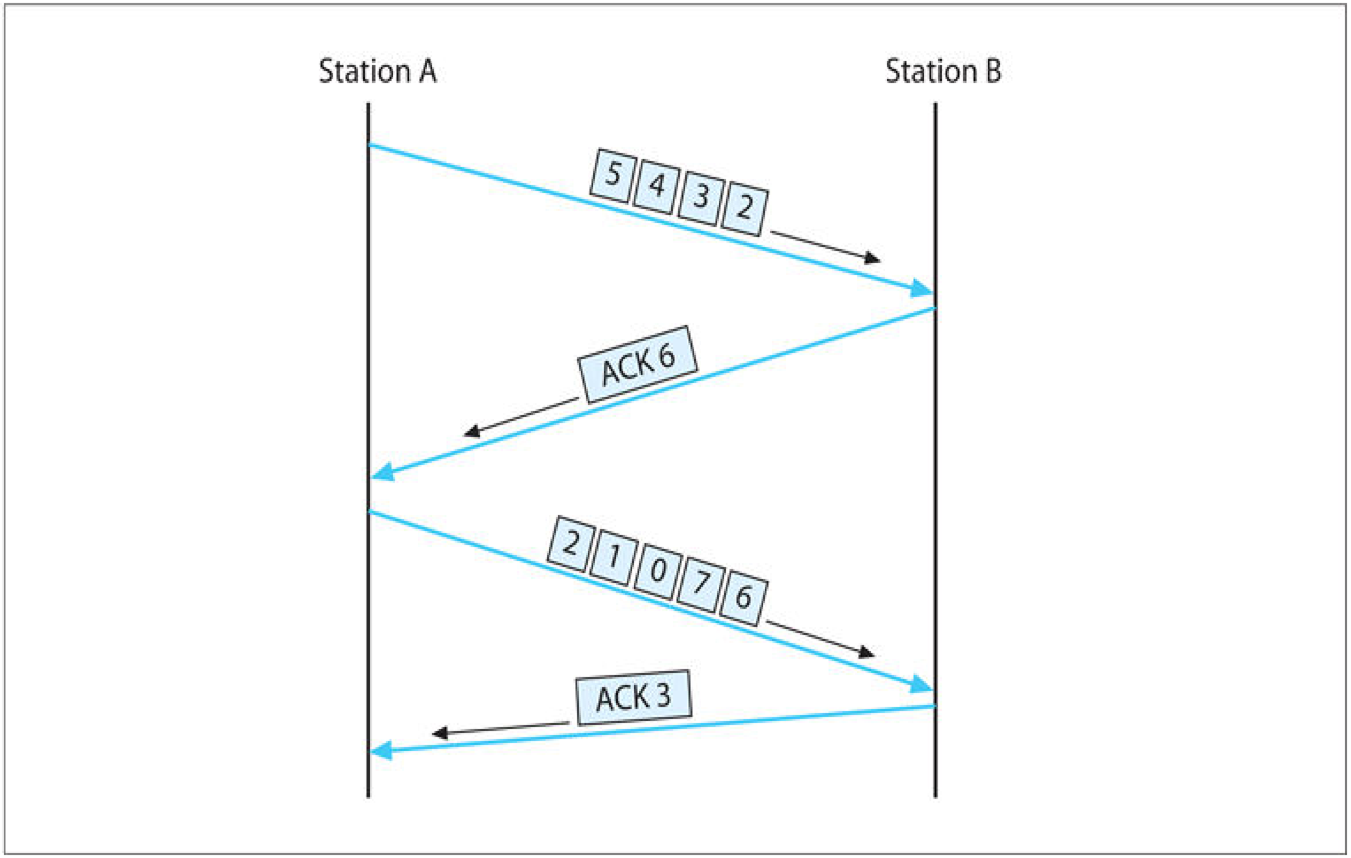


Sliding Window Error Control (continued)

- Older sliding window protocols numbered each frame or packet that was transmitted
- More modern sliding window protocols number each byte within a frame
- An example in which the packets are numbered, followed by an example in which the bytes are numbered:

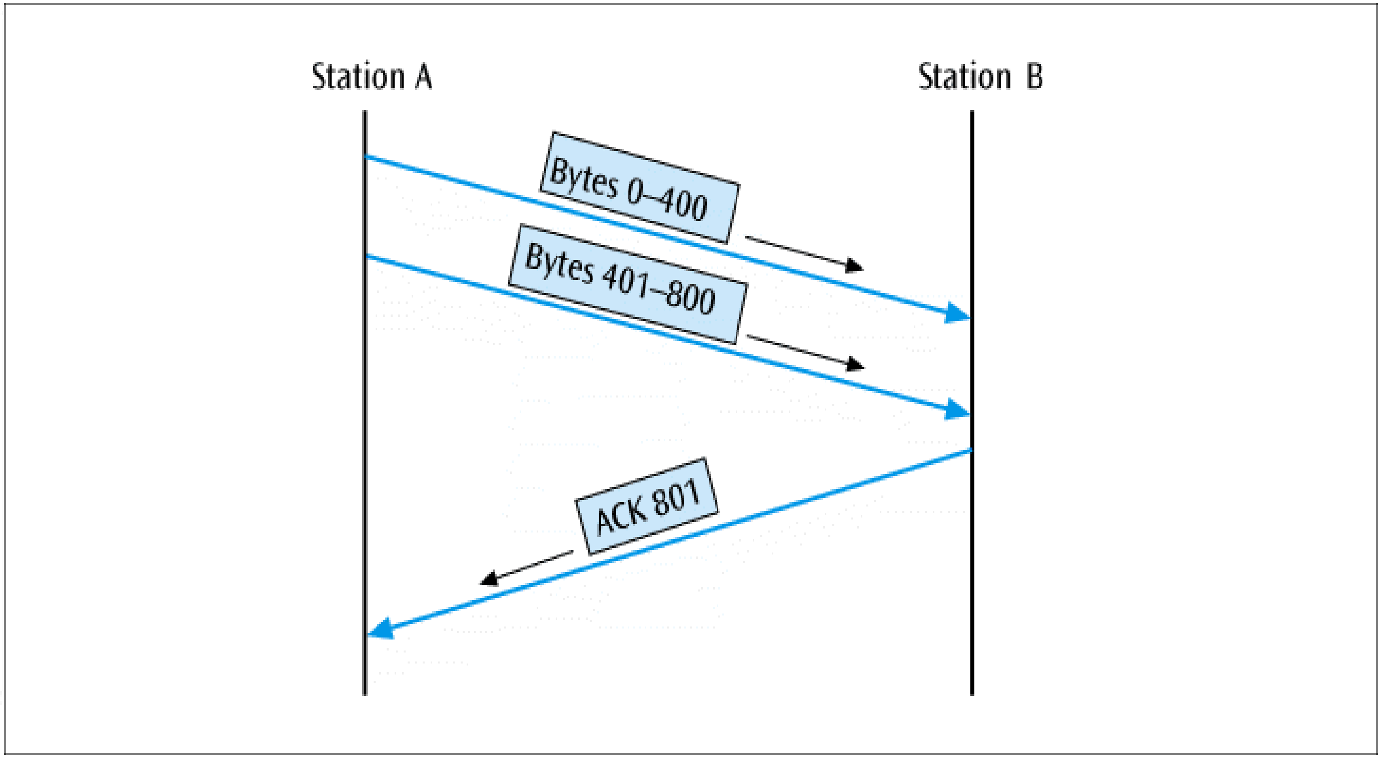
Sliding Window Error Control (continued)

Figure 6-10
Normal transfer of data between two stations with numbering of the packets



Sliding Window Error Control (continued)

Figure 6-11
Normal transfer of data between two stations with numbering of the bytes



Sliding Window Error Control (continued)

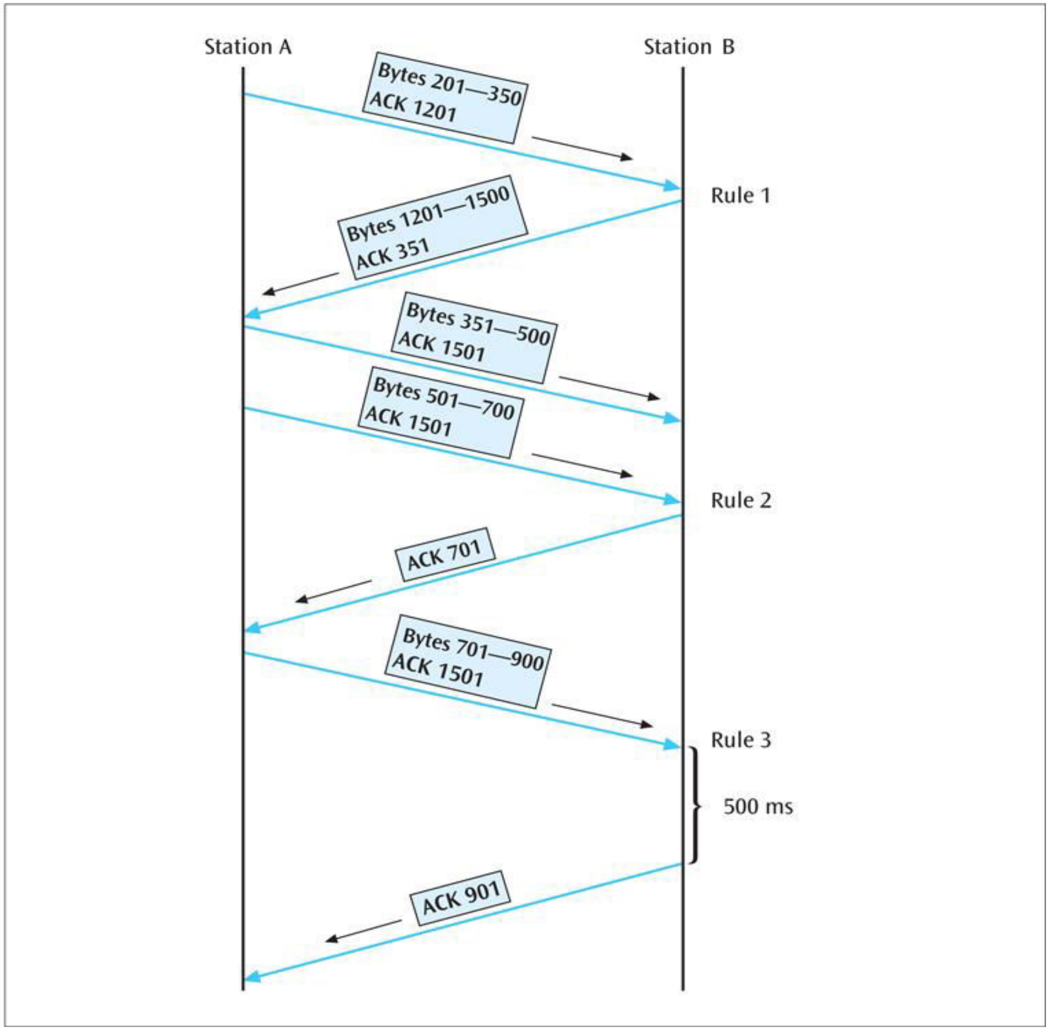
- Notice that an ACK is not always sent after each frame is received
  - It is more efficient to wait for a few received frames before returning an ACK
- How long should you wait until you return an ACK?

Sliding Window Error Control (continued)

- Using TCP/IP, there are some basic rules concerning ACKs:
  - Rule 1: If a receiver just received data and wants to send its own data, piggyback an ACK along with that data
  - Rule 2: If a receiver has no data to return and has just ACKed the last packet, receiver waits 500 ms for another packet
    - If while waiting, another packet arrives, send the ACK immediately
  - Rule 3: If a receiver has no data to return and has just ACKed the last packet, receiver waits 500 ms
    - No packet, send ACK

Sliding Window Error Control (continued)

Figure 6-12
Three examples of returning an acknowledgment (ACK)

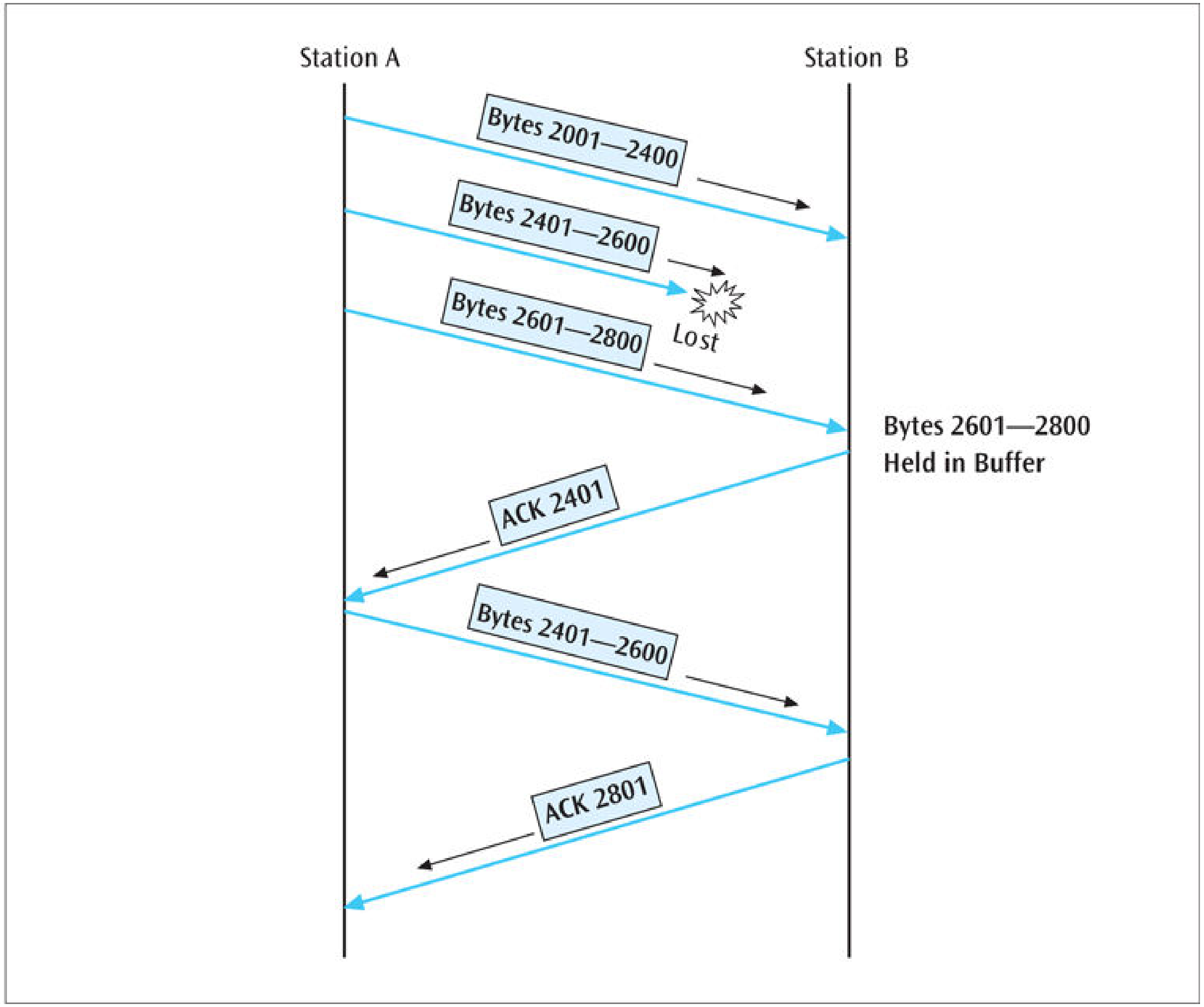


Sliding Window Error Control (continued)

- What happens when a packet is lost?
  - As shown in the next slide, if a frame is lost, the following frame will be “out of sequence”
    - The receiver will hold the out of sequence bytes in a buffer and request the sender to retransmit the missing frame

Sliding Window Error Control (continued)

Figure 6-13
A lost packet and Station B's response

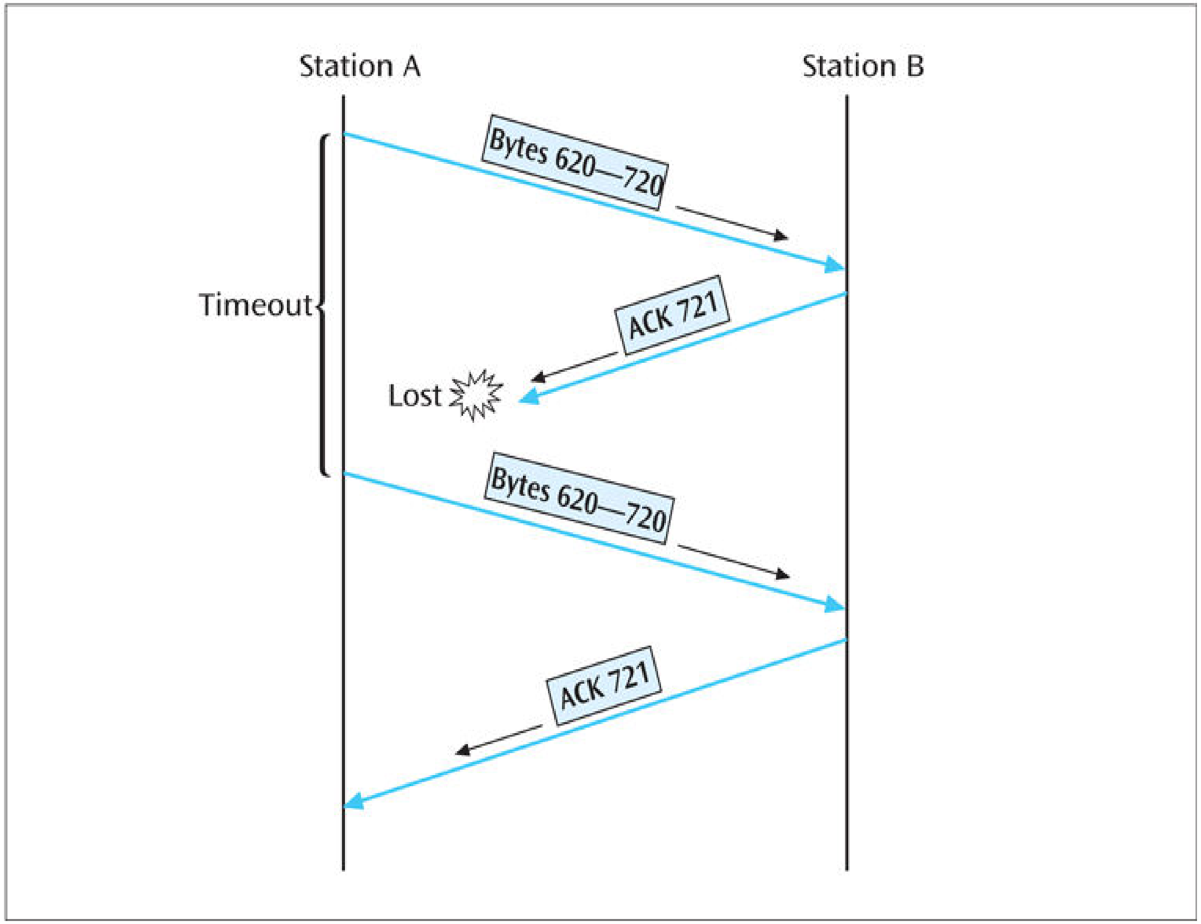


Sliding Window Error Control (continued)

- What happens when an ACK is lost?
  - As shown in the next slide, if an ACK is lost, the sender will wait for the ACK to arrive and eventually time out
    - When the time-out occurs, the sender will resend the last frame

Sliding Window Error Control (continued)

Figure 6-14
A lost acknowledgment and the retransmission of a packet



Correct the Error

- For a receiver to correct the error with no further help from the transmitter requires a large amount of redundant information to accompany the original data
  - This redundant information allows the receiver to determine the error and make corrections
- This type of error control is often called forward error correction and involves codes called Hamming codes

Correct the Error (continued)

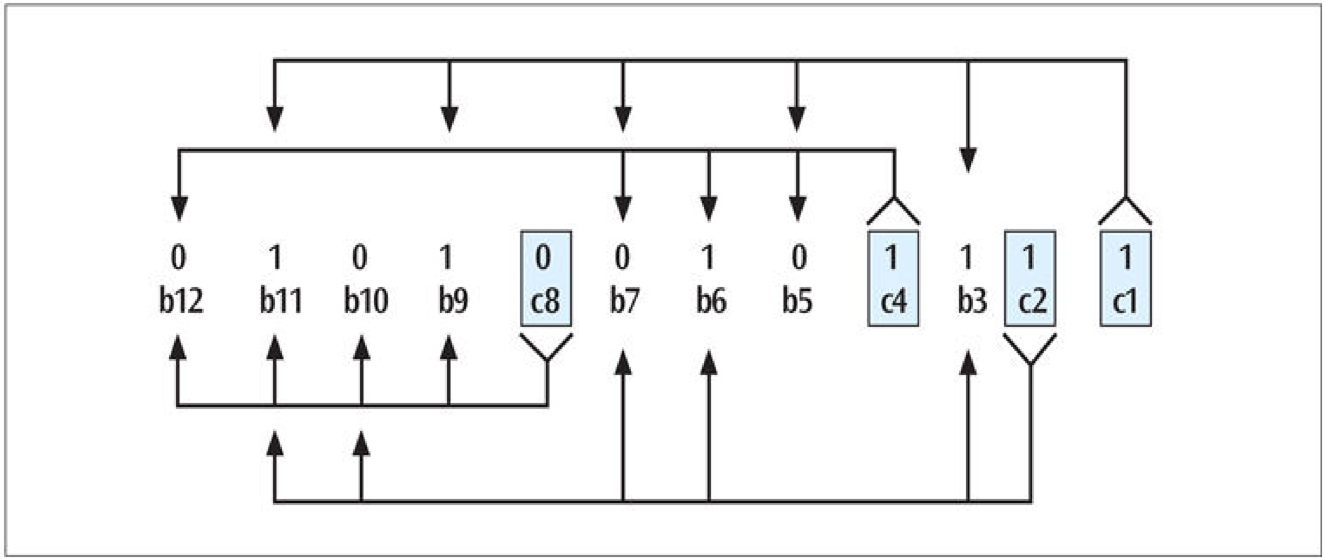
- Hamming codes add additional check bits to a character
  - These check bits perform parity checks on various bits
- Example: One could create a Hamming code in which 4 check bits are added to an 8-bit character
  - We can number the check bits c8, c4, c2 and c1
  - We will number the data bits b12, b11, b10, b9, b7, b6, b5, and b3
  - Place the bits in the following order: b12, b11, b10, b9, c8, b7, b6, b5, c4, b3, c2, c1

Correct the Error (continued)

- Example (continued):
  - c8 will perform a parity check on bits b12, b11, b10, and b9
  - c4 will perform a parity check on bits b12, b7, b6 and b5
  - c2 will perform a parity check on bits b11, b10, b7, b6 and b3
  - c1 will perform a parity check on bits b11, b9, b7, b5, and b3
- The next slide shows the check bits and their values

Correct the Error (continued)

Figure 6-15
Hamming code check bits generated from the data 01010101



Correct the Error (continued)

- The sender will take the 8-bit character and generate the 4 check bits as described
  - The 4 check bits are then added to the 8 data bits in the sequence as shown and then transmitted
- The receiver will perform the 4 parity checks using the 4 check bits
  - If no bits flipped during transmission, then there should be no parity errors
- What happens if one of the bits flipped during transmission?

Correct the Error (continued)

- For example, what if bit b9 flips?
  - The c8 check bit checks bits b12, b11, b10, b9 and c8 (01000)
    - This would cause a parity error
  - The c4 check bit checks bits b12, b7, b6, b5 and c4 (00101)
    - This would not cause a parity error (even number of 1s)
  - The c2 check bit checks bits b11, b10, b7, b6, b3 and c2 (100111)
    - This would not cause a parity error

Correct the Error (continued)

- For example, what if bit b9 flips? (continued)
  - The c1 check bit checks b11, b9, b7, b5, b3 and c1 (100011)
    - This would cause a parity error
  - Writing the parity errors in sequence gives us 1001, which is binary for the value 9
    - Thus, the bit error occurred in the 9th position

Error Detection In Action

- FEC is used in transmission of radio signals, such as those used in transmission of digital television (Reed-Solomon and Trellis encoding) and 4D-PAM5 (Viterbi and Trellis encoding)

- Some FEC is based on Hamming Codes

Summary

- Noise is always present in computer networks, and if the noise level is too high, errors will be introduced during the transmission of data
  - Types of noise include white noise, impulse noise, crosstalk, echo, jitter, and attenuation
- Among the techniques for reducing noise are proper shielding of cables, telephone line conditioning or equalization, using modern digital equipment, using digital repeaters and analog amplifiers, and observing the stated capacities of media

Summary (continued)

- Three basic forms of error detection are parity, arithmetic checksum, and cyclic redundancy checksum
- Cyclic redundancy checksum is a superior error-detection scheme with almost 100 percent capability of recognizing corrupted data packets
- Once an error has been detected, there are three possible options: do nothing, return an error message, and correct the error

Summary (continued)

- Stop-and-wait protocol allows only one packet to be sent at a time
- Sliding window protocol allows multiple packets to be sent at one time
- Error correction is a possibility if the transmitted data contains enough redundant information so that the receiver can properly correct the error without asking the transmitter for additional information