Chirag solanki

Assignment 10

15035

Answer

R2

No, even if all the individual links were completely reliable, this would not guarantee that the end-to-end communication between hosts was reliable. For example, individual routers or switches might drop packets for forwarding from one link to another, the sending host's network interface card might drop the packet, or the receiving host's network interface card might lose the packet. The reliability at the link-layer, between point-to-point nodes on the Internet only guarantees local reliability. To provide end-to-end reliability between two arbitrary hosts requires a higher layer service.

R10

C’s adapter will process the frames, but the adapter will not pass the datagrams up the protocol stack. If the LAN broadcast address is used, then C’s adapter will both process the frames and pass the datagrams up the protocol stack.

Essay:

A checksum or hash sum is a small-size datum from an arbitrary block of digital data for the purpose of detecting errors which may have been introduced during its transmission or storage. It is usually applied to an installation file after it is received from the download server. By themselves checksums are often used to verify data integrity , but should not be relied upon to also verify data authenticity .

Checksums take on various forms, depending upon the nature of the transmission and the needed reliability. For example, the simplest checksum is to sum up all the bytes of a transmission, computing the sum in an 8-bit counter. This value is appended as the last byte of the transmission. The idea is that upon receipt of *n* bytes, you sum up the first *n-*1 bytes, and see if the answer is the same as the last byte. Since this is a bit awkward, a variant on this theme is to, on transmission, sum up all the bytes, the *negate* the checksum byte before transmitting it. This means that the sum of all *n* bytes should be 0. These techniques are not terribly reliable; for example, if the packet is known to be 64 bits in length, and we receive 64 '\0' bytes, the sum is 0, so the result must be correct. Of course, if there is a hardware failure that simply fails to transmit the data bytes (particularly easy on synchronous transmission, where no "start bit" is involved), then the fact that you receive a packet of 64 0 bytes with a checksum result of 0 is misleading; you think you've received a valid packet and you've received nothing at all. A solution to this is to do something like negate the checksum value computed, subtract 1 from it, and expect that the result of the receiver's checksum of the *n* bytes is 0xFF (-1, as a signed 8-bit value). This means that the 0-lossage problem goes away.

Nonetheless, for all its simplicity, the checksum technique just described is remarkably weak. For example, if we were to transpose two of the characters of the transmission, the result would be the same, so although the wrong packet is received, a correct checksum is believed. Certain kinds of noise injection on the line can also introduce undetectable errors because the noise that mangles one byte is cancelled by the noise that mangles another byte.

Therefore, my techniques generally are useful when a few thousand bytes of state are involved, such as in a dialog.

I use a technique that has no particular theoretical justification. But I've found it to be reliable for my purposes. The story is that I wanted to use CRC-32 some years ago, but couldn't locate the source code for a CRC-32 algorithm on the Web at that time, so I turned to my Adobe Type 1 Font Handbook and cribbed their encryption algorithm. But rather than encrypt the data, I just used the basic algorithm to create a 32-bit checksum..

Code:

Screenshot:

#include <stdio.h>

unsigned char checksum (unsigned char \*ptr, size\_t sz) {

unsigned char chk = 0;

while (sz-- != 0)

chk -= \*ptr++;

return chk;

}

int main(int argc, char\* argv[])

{

unsigned char x[] = "Hello\_";

unsigned char y = checksum (x, 5);

printf ("Checksum is 0x%02x\n", y);

x[5] = y;

y = checksum (x, 6);

printf ("Checksum test is 0x%02x\n", y);

return 0;

}

Output

Checksum is 0x0c

Checksum test is 0x00