Complementary Lecture Notes on Computer Networks (WIP)
Chapter 1

(C) 2013, 2014 José María Foces Morán, lecturer. University of León

April 4, 2014

Revision: ~~V.1.2~~

Introduction

These notes aim to complement the lectures of the undergraduate course on Computer Networks. Hereon you will find outstanding concepts commented and exercises illustrated.

Clarifications about network performance

Latency and bandwidth are two important performance measures used in computer systems and in computer networks. Latency is how much time it takes to perform an operation and bandwidth is a rate, the ratio of the number of operations performed and the time they took. Let's consider how much time it takes to transfer one bit of information from a computer A that is directly connected to computer B.

The first component of the latency in the case of *direct connection* is the propagation delay Tp or the time it takes for an electromagnetic signal to propagate from A to B through the considered communication medium. Each physical communication medium propagates electromagnetic signals at a fraction of c, the speed of light in empty space, this fraction is a property of that medium. Propagation delay is also known as *one-way delay*. Bandwidth has a strong dependency on the physical properties of the communication medium, it is the standardized width of that medium's *transfer function*: a bigger bandwidth will allow higher transmission speeds as we will study in the chapter 2 section devoted to the Shannon-Hartley theorem.

Another important performance measure, derived from Tp is the round-trip time or RTT. Assuming that the propagation times are the same in both directions, then $RTT = T_p + T_p$. Alongside with bandwidth, this performance measure plays an important role in the design of end-to-end reliable transfer protocols such as TCP. Specifically, if we want to attain a maximum utilization of the network we will have to maximize the product delay x bandwidth. Depending on the specific circumstances, that delay can be the one-way delay or the RTT.

Fig. 1 graphically illustrates these concepts. On computer A we need to transfer one bit of information to computer B. The first we need to do is to copy that bit from memory to the signalling electronics on the network interface which will translate it into a waveform. The vertical red line represents the time it takes to perform this translation of a binary symbol into a waveform, we refer to this time as *transmission time*. Intuitively, you will readily see that longer transmission times will negatively affect the transmission speed that we can *safely* attain, conversely, if you attempt to transmit with increasingly shorter transmission times, the resulting waveforms will not faithfully represent the original bits of information upon arrival at B. What is the limit to how fast we can transform bits into waveforms? We will study these factors with sufficient detail in ch. 2, for the time being, it will suffice to say that the tranmission medium bandwidth limits our ability to transmit with increasingly shorter times. Shorter tranmission times or higher transmission speeds demand higher bandwidth; bandwidth is physically limited, always.

Continuing with Fig.1, once we have successfully transmitted one bit, the resulting waveform begins to propagate through the medium, this is represented by the sky blue arrow. How long does it take for the waveform to propagate from A to B? Obviously it depends directly on the length of the link and inversely on the propagation speed. The sum of the transmission time (red) and the propagation time (sky blue) is the latency: how long it takes for a bit to be transferred from A to B. As you can see, latency in this example has the two components mentioned: transmission time and propagation time.

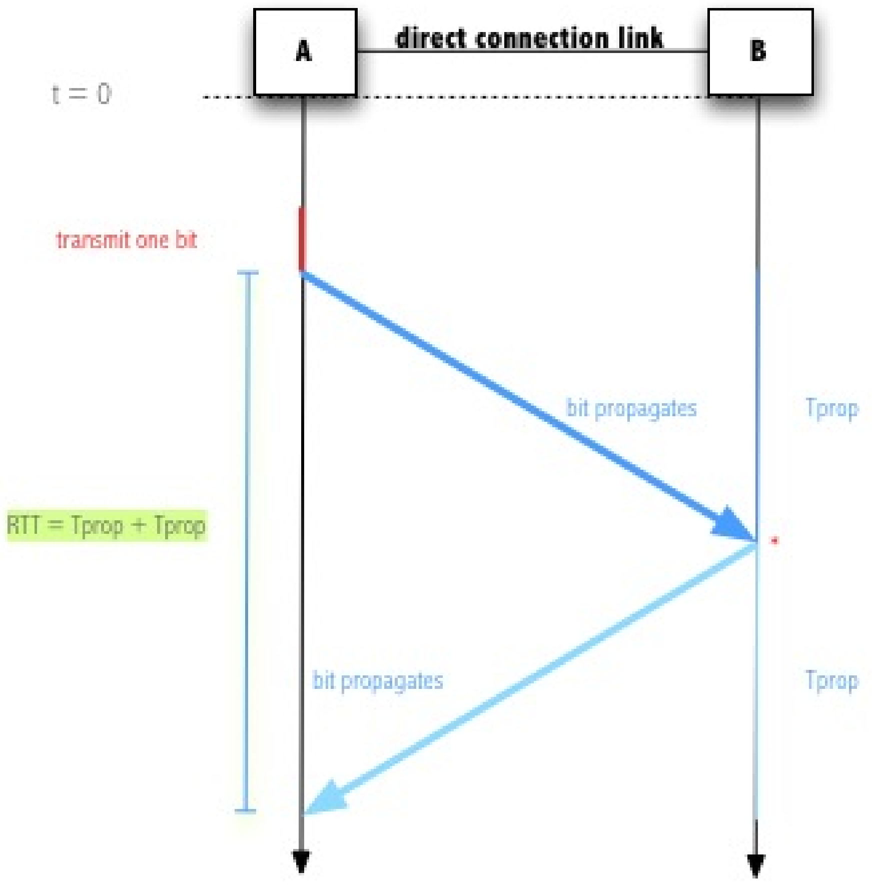
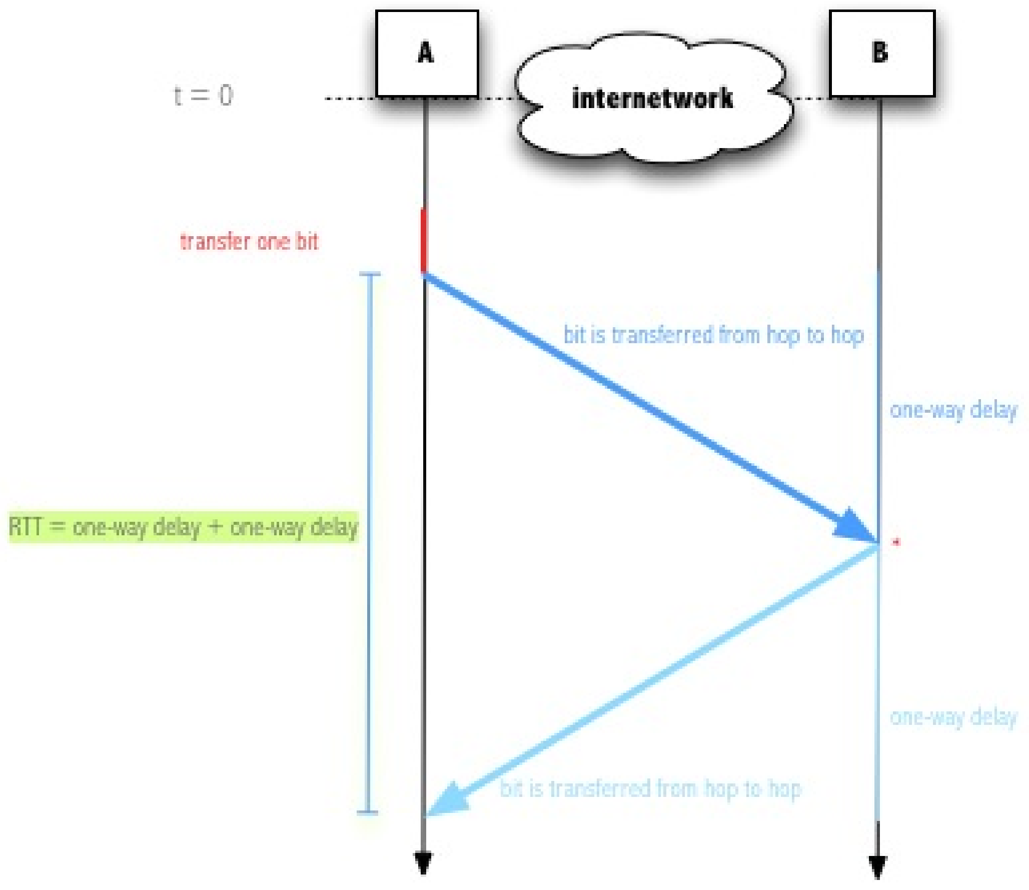


Fig. 1. Latency in transferring one bit from A to B when they are directly connected

In the context of a non-direct connection between A and B, these concepts receive different names and the definitions themselves are different than those that apply in the direct-connection case. Many times they are used interchangeably. When the two hosts considered are not directly connected, we assume that they communicate over an internetwork and, therefore, the channel in between is not a *physical* one, but a *logical* one.

Let's study the case of a logical channel between A and B, i.e., an arbitrary interconnection of links. In this case, the equivalent to the propagation time is named one-way delay, the time it takes for one bit to be transferred from A to B; bandwidth in this case is named throughput and it is the effective maximum transmission speed (bits/sec) *achievable* from end to end (A -> network -> B).

The round-trip time of a logical channel is an essential parameter in the operation of certain protocols, TCP, for example. Let's assume that A and B are connected by a logical channel, then, the RTT is the sum of the two one-way delays: $RTT = delay_{one-way} + delay_{one-way}$. Normally, for the transmission of *one bit* at A (red) the transmission time will be much smaller than the delay, thus, the biggest contribution to latency will come from the one-way delay, not from the transmission time. In the problems explained in these notes, we will see that when a big amount of data is transferred from A to B, the one-way delay due to one bit's propagation is hidden in the tranmission times of the bits that are to be transmitted right after the one that is propagating. In conclusion, if several transmission times fit within a one-way delay, when transferring a big amount of data it's the transmission times that account for the total transfer time, consequently, the one-way delay is negligible.


A
internetwork
B
t = 0
transfer one bit
bit is transferred from hop to hop
one-way delay
RTT = one-way delay + one-way delay
one-way delay
bit is transferred from hop to hop

Mesauring the RTT

To measure the RTT (End-to-end RTT) between a client thread (A) and a server thread (B), according to the above definition, we will have to guarantee that the time between the reception of the incoming *bit* at B and it bouncing back a reponse bit directed towards A, should be reduced to a minimum, ideally 0 (See the red-colored * in the latter figure). This amounts to guaranteeing in thread B that we perform *virtually* no computation between receiving the bit and sending the response back, thereby reducing its time length as much as possible, i.e. imposing the least influence on the RTT.

Exercise 4

Calculate the total time required to transfer a 1.5 MB file in the following cases, assuming an RTT of 80ms, a packet size of 1 KB data, and an initial 2ŒRTT of "handshaking" before data is sent:

- (a) The bandwidth is 10 Mbps, and data packets can be sent continuously
  In order to simplify the exercise we will assume a direct connection between A and B, therefore, in computing the results we will be using bandwidths and propagation times. A and B are computer systems, A has a 1.5MB file that is to be transferred to B and we are requested to compute the whole amount of time it takes to transfer the file. That the bandwidth is 10 Mbps is to be understood in the sense that we are allowed to transmit at that speed, for, in fact we could have a 10 Mbps channel and transmit at a speed of 500 Kbps. Therefore we are using a transmission speed of 10 Mbps, we denote that speed as bandwidth BW = 10 Mbps.

  Since in (a) packets can be sent in a continuous fashion, i.e., with no pause in between, in fact we will transmit the whole bunch of bits (1.5MB) in a single, big block. The exercise also specifies that, before any bit can be transmitted, a handshake takes place which consumes a length of time of 2 x RTT. For the time being, the handshake is an initial series of information interchanges by which transmitter and receiver fix some communication protocol parameters before proceeding to the file transfer itself. We will delve into more detail regarding that handshake in ch. 2 and others. After the handshake, A begins transferring the first bit and the rest at the speed specified in the bandwidth. Observe that the first bit arrives at B Tprop seconds after it was transmitted at A. Observe also that, while bit 1 is being propagated, bit 2 begins transmission/propagation and the same with the rest of bits (Other subsequent bits transmission times could be hidden in the propagation time of one bit, depending on the ratio of transmission time to propagation time at use). Figure 3 is a representation of the times mentioned. The total time will be the sum of all the times in between the start of the handshake at B and the reception of the last bit from the 1.5MB file:

$$T_{total} = 2 \times RTT + N_{bits} \times T_{transm} + T_{prop}$$

Let's convert the file size of 1.5MB to bits:

$$N_{bits} = 1.5MB = 1.5 \times 2^{20} Byte \frac{8bits}{Byte} = 12 \times 2^{20} bits$$

and calculate the transmission time it takes to transmit a single bit:

$$T_{transm} = \frac{1}{BW \frac{bits}{sec}} = \frac{1}{BW} \frac{sec}{bit} = \frac{1}{10 \times 10^6} \frac{sec}{bit} = 0.1 \times 10^{-6} \frac{sec}{bit}.$$

Now, we can evaluate the total time consumed by the continuous 1.5MB transfer:

$$T_{total} = 2 \times 80 \times 10^{-3} sec + 12 \times 2^{20} bits \times 0.1 \times 10^{-6} \frac{sec}{bit} + \frac{80}{2} \times 10^{-3} sec = 1.4582 sec$$

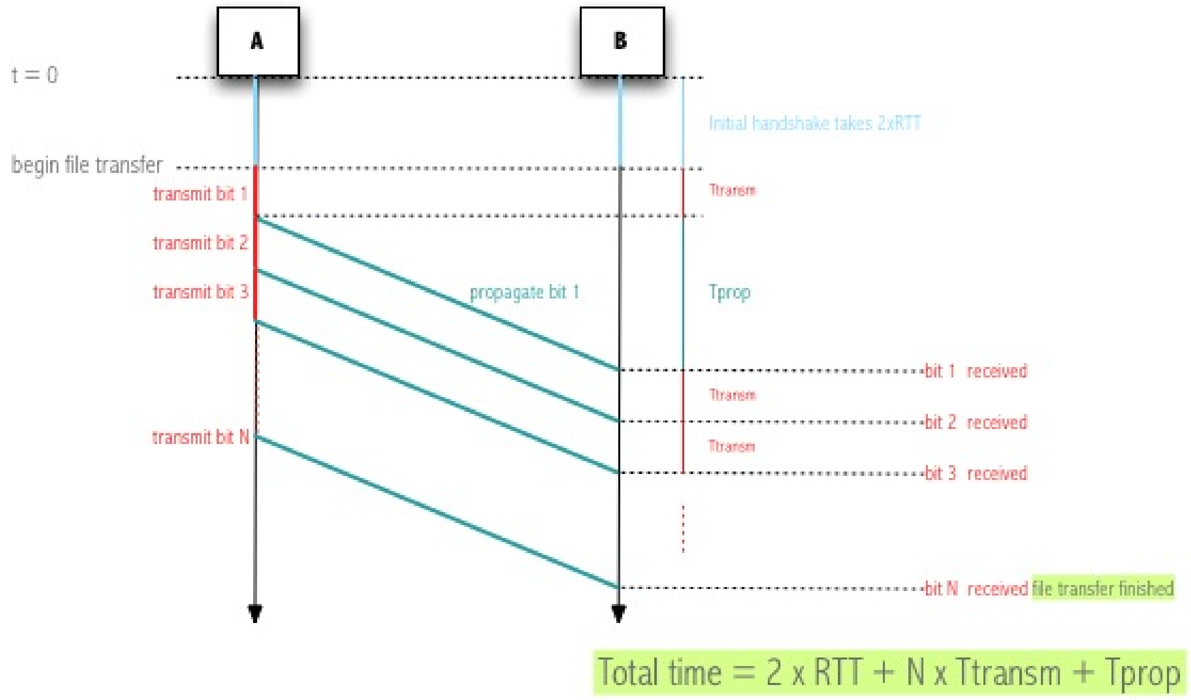


Fig. 3. Transmission and propagation times in a direct link

- (b) Bandwidth is 10 Mbps, but after sending each data packet, the transmitter must wait one RTT before sending the next one. Fig. 4 contains a timeline representation of this exercise section, the initial handshake is light blue at the top, then transfer of of first packet begins. By contrast with section (a), packet transfer is not continuous, after each packet *the transmitter* inserts a pause the length of an RTT (Represented in orange). Inspecting the timeline on Fig. 4, we can derive the total time:

  $T_{total} = 2 \times RTT + (N_{pack} - 1) \times (T_{1pack} + \frac{RTT}{2}) + T_{1pack}$

  where T_{1pack} represents the time necessary to *transfer* a packet *(a.k.a. Packet Transfer Latency)* and N_{pack} is the total number of packets contained in the 1.5MB (This section specifies that the transmitter will packetize the information before *transferring* it using a packet size of 1KB). Now, we can calculate the number of packets:

  $N_{pack} = \frac{FileSize}{PacketSize} = \frac{F_s}{P_s} = \frac{1.5MB}{1KB} = 1.5\frac{2^{20}}{2^{10}} = 1.5 \times 2^{10} packets = 1536 packets$

  For calculating the time necessary to *transfer* a full packet we will study the timeline in fig. 4 (T_{1pack} includes the *transmission* times of all the bits belonging to the packet plus the propagation time of the first bit). Observe that, effectively, after receiving each packet, reception of packets' bits is paused for $\frac{RTT}{2} sec$, not the full RTT that we might expect, the reason is that the propagation of the last bit of a packet is concurrent with the first half of the RTT pause inserted by the transmitter between any two back-to-back packets. According to the timeline on the right of fig. 4 we have:

  $T_{1pack} = T_{transm} + T_{prop} + (N_{bits-packet} - 1) \times T_{transm}$

  $T_{1pack} = N_{bits-packet} \times T_{transm} + T_{prop}$

  Let's begin by calculating the number of bits per packet:

  $N_{bits-packet} = 1\frac{KB}{packet} \times \frac{2^{10}}{1K} \times \frac{8bits}{1B} = 8192\frac{bits}{packet}$

  No, we can apply the formula above to calculate the packet transfer latency:

  $T_{1pack} = 8192\frac{bits}{packet} \times 0.1 \times 10^{-6}\frac{sec}{bit} + \frac{80}{2} \times 10^{-3} sec_{prop} = 0.040819 sec$

  Finally, we calculate the total time taken by the receiver to collect the 1.5MB; as in the previous case, the file transfer will start with the handshake, then, the first packet will be transferred, bit by bit and introduce a pause of length $2 \times RTT$ and repeat the process through the last packet (Obviously, the $2 \times RTT$ pause is not necessary after the last packet).

$T_{total} = 2 \times 80 \times 10^{-3} + (1536 - 1) \times (0.040819 + 40 \times 10^{-3}) + 0.040819$

$T_{total} = 124.2579 sec$

Before finishing this section, we might want to quantify the influence of the inter-packet pauses on the overall performance attained in this case, for, we are sure that the throughput (The effective transfer rate achieved) in this case (b) must necessarily be less than in case (a). The throughput results of the ratio: total file size transferred by the total time just calculated. Let's calculate it:

$Throughput = \frac{N_{bits}}{T_{total}} = \frac{12 \times 2^{20} bits}{124.2579 sec} = 101264.483 bps = 101.264 Kbps$

What do you think about the result? Is the throughput rather low? The bandwidth is 10Mbps and throughput represents barely a $\frac{101.264 Kbps}{10 Mbps} \times 100\% \cong 1\%$

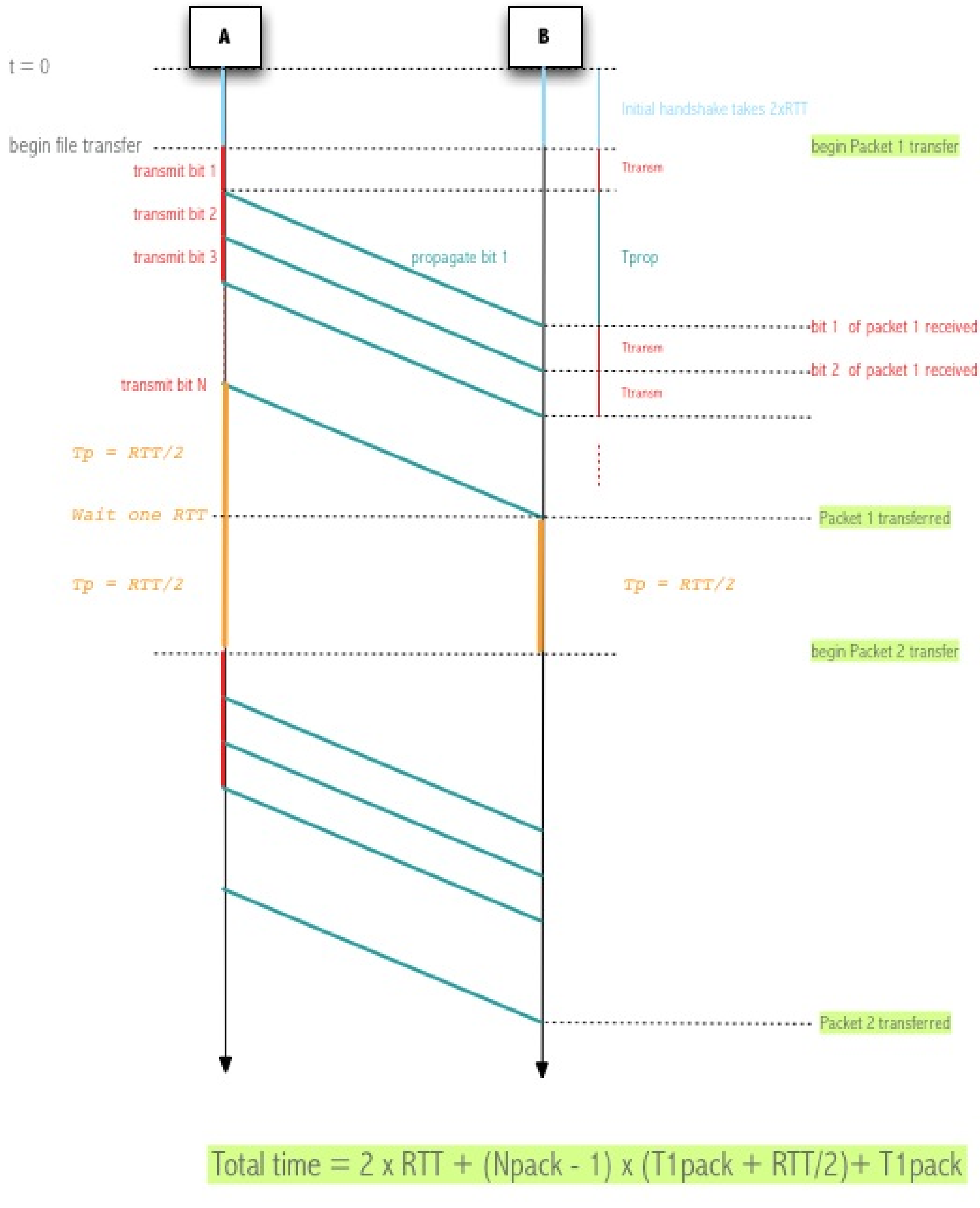


Fig. 4. A packetized transfer with a pause in between two consecutive packets

- (c) The link allows infinitely fast transmit (That means infinite bandwidth), but limits *throughput* (not bandwidth) such that only 20 packets can be sent per RTT

This exercise statement means that we can transmit as fast as we can but, in fact, the medium will only accept 20 packets sent within a time of RTT seconds, that can be summarized by stating the the *maximum attainable throughput* is the number of bits per second resulting from:

$Throughput_{max} = \frac{20}{80 \times 10^{-3}} \frac{packets}{sec}$ or

$Throughput_{max} = \frac{20}{80 \times 10^{-3}} \frac{packets}{sec} \cdot 8192 \frac{bits}{packet} = 2.048 \times 10^{6} \frac{bits}{sec}$

In this case, as in case (a), the transmission is continuous, i.e., the 1.5MB are transferred from A to B without pause, however, by contrast to (a) the attainable throughput is $Throughput_{max}$, then the calculations are the same as in (a).

- (d) Zero transmit time as in (c), but during the first RTT we can send one packet, during the second RTT we can send two packets, during the third we can send four (231), etc. (A justification for such an exponential increase will be given in Chapter 6.)
  In this case throughput is being increased at each succeding packet according to a multiplicative law, therefore, in order for you to calculate the total time you will have to calculate the number of packets sent at each time instant which is governed by a geometric progression. We recommend you to complete the calculations and check the result with the textbook soutions section. Also, for the time being, you can assume that the network is completely stable and that therefore, the *RTT is constant* (In general, this is not true, for, when the offered load is over some threshold, the queuing delays and packet loss will grow fast which will cause the RTT to grow unbounded).

Exercise 14

Suppose a 128-kbps point-to-point link is set up between the Earth and a rover on Mars. The distance from the Earth to Mars (when they are closest together) is approximately 55 Gm, and data travels over the link at the speed of light—$3 \times 10^{8} \frac{m}{s}$.

- a) Calculate the minimum RTT for the link

  The RTT is the Round-trip-time, or the time it takes for 1 bit to travel to the destination and then back to the sender, normally we assume that $RTT = 2 \times T_p =$

$2 \times delay_{one-way}$ according to the definitions given in the textbook ch.1 and the introductory sections of this document. The requested minimum RTT will occur when the distance from the Earth to Mars is minimum:

Let's denote the speed of light in empty space c, then: $RTT_{min} = 2 \cdot \frac{d_{min-earth-mars}}{c} = \frac{2 \cdot 55 \times 10^9 m}{3 \times 10^8 \frac{m}{sec}} = 2 \cdot 183.33 sec = 366.66 sec$

- b) Calculate the delay x bandwidth product for the link

  Recall the significance of this product: If we want to maximize network utilization, the number of bits that should be in-flight from A to B before the first of them has arrived at B, is the delay x bandwidth product. Delay in this case is the one-way delay, but, normally when speaking of the delay x product, delay is the RTT, not the one-way delay. In all exercises we will make clear which definition of delay must be used: either the one-way delay or the RTT.

  $delay \times bandwidth = delay_{one-way} \times bandwidth = \frac{RTT}{2} \times BW$

  $183.33 sec \times 128K \frac{bits}{sec} \cdot \frac{10^3}{1K} = 23466240 bits \cdot \frac{1B}{8bits} \cdot \frac{1M}{2^{20}} = 2.7974 MB$

- c) A camera on the rover takes pictures of its surroundings and sends these to Earth. How quickly after a picture is taken can it reach Mission Control on Earth? Assume that each image is *5MB* in size (The textbook's exercise specifies *5Mb* which, according to the textbook solution, is a mistake)

  The transfer process of each image is continuous, i.e., no packetization is at play here. The transfer scheme is the same as that used in exercise 4 section (a), we will perform the simple calculation according to that scheme:

  $T_{total} = delay_{one-way} + T_{transm-5Mbits}$

  The transmission time (Corresponding to a bandwidth of 128-kbps) is:

  $T_{transm-1bit} = \frac{1}{Bw} sec = \frac{1bit}{128 \times 10^3 \frac{bit}{sec}} = 7.81 \times 10^{-6} sec$

  $T_{transm-5Mbits} = 5 \times 2^{20} Byte \cdot \frac{8bits}{1Byte} \cdot T_{transm-1bit} =$

  $40 \times 2^{20} bit \cdot 7.81 \times 10^{-6} \frac{sec}{bit} = 327.575 sec$

  $delay_{one-way = \frac{RTT_{min}}{2} = \frac{366.66}{2} = 183.33 sec}$

$T_{total} = 183.33 + 327.575 = 510.909 sec$

Exercise 17

Calculate the latency (from first bit sent to last bit received) for:

- (a) 1-Gbps Ethernet with a single store-and-forward switch in the path and a packet size of 5000 bits. Assume that each link introduces a propagation delay of 10 s and that the switch begins retransmitting immediately after it has finished receiving the packet.
  $N_{bits-packet} = 5000$ bits

  $Delay_{one-way} = T_p = 10 \times 10^{-6} sec$

  $BW = 1\, Gbps$ por tanto, el tiempo de transmisión de 1 bit es:

  $T_{transm} = \frac{1}{BW} \frac{sec}{bit} = \frac{1}{1 \times 10^9} \frac{sec}{bit} = 10^{-9} \frac{sec}{bit}$

  $T_{total} = T_p + T_{transm} \cdot 5000 + T_p + T_{transm} \cdot 5000 = 2 \cdot 5000 \times 10^{-9} + 2 \cdot T_p =$
  $10 \cdot 10^3 \times 10^{-9} + 2 \cdot 10 \times 10^{-6} = 3 \cdot 10 \times 10^{-6} = 30 \mu sec$

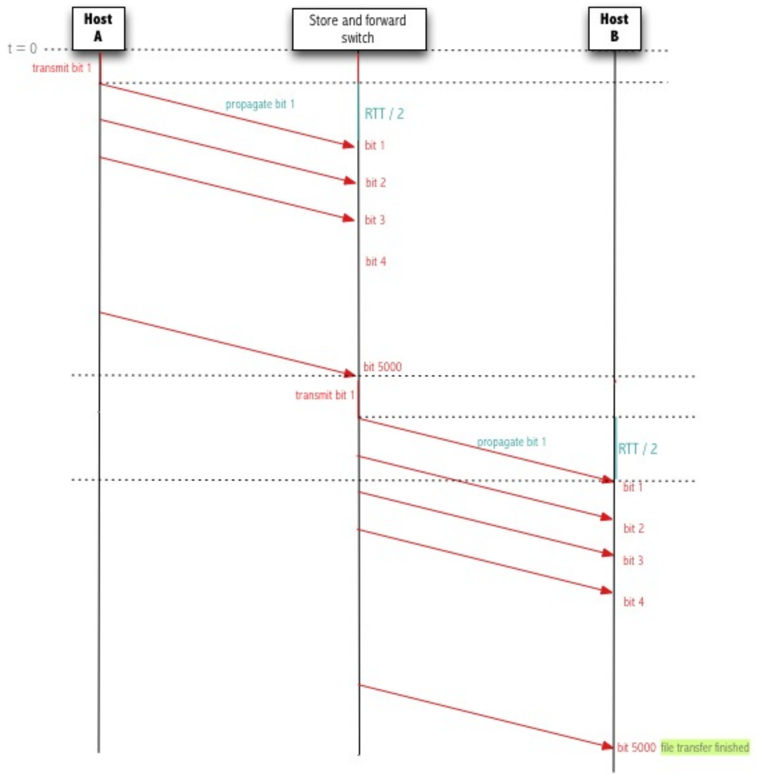


Fig. 5. Transfer of one packet from Host A to host B via a store-and-forward switch

- (b) Same as (a) but with three switches.
  Insert two additional swtiches, since the time cost of each packet transfer is $15\mu sec$, in this case, with three switches between A and B, we will have four transfers, a total of $4 \times 15\mu sec = 60\mu sec$.

Exercise 27

For the following, as in the previous exercise (No. 26), assume that no data compression is done. Calculate the bandwidth necessary for transmitting in real time:

- (a) High-definition video at a resolution of 1920x1080, 24 bits/pixel, 30 frames/second
  The video is represented a series of ordered frames that are drawn on the screen at a rate of 30 frames/second so that a person may perceive the movement of the scenes. Each frame contains 1920 bits on the horizontal axis and 1080 bits on the vertical axis, thus, the total number of bits contained in a single frame will be:

$1920 \times 1080 \frac{pixel}{frame} = 2073600 \frac{pixel}{frame}$

Each of those pixels, according to the problem statement, is represented by 24 bits, i.e., 3 bytes, each of them represents a color level in the RGB color representation system for displays. The total frame size, in bits, is:

$F_{size} = 2073600 \frac{pixel}{frame} \cdot 24 \frac{bits}{pixel} = 49766400 \frac{bit}{frame}$

the bandwidth is $30 \frac{frame}{sec}$, therefore the bandwidth expressed in bits/sec will be:

$30 \frac{frame}{sec} \cdot 49766400 \frac{bit}{frame} = 1.493 Gbps$

- (b) POTS (plain old telephone service) voice audio of 8-bit samples at 8 KHz

  The telephone user voice is digitzed at standardized rate of 8KHz, each resulting voice sample, in this case is represented by using 8 bits. In the previous case, colors were represented by using 24 bits, a much higher resolution, it looks as though our eye sight can resolve much more detail than the ears can with sound. Perform the calculations as in (a).

- (c) GSM mobile voice audio of 260-bit samples at 50 Hz

  This case corresponds to the bandwidth consumed by a GSM cellular phone: higher resolution in each voice sample but at a rather low frequency. In chapter 2 we will introduce some basic theorems that will shed some light on the topics related to digitization quality and channel utilization. Proceed in a manner similar to (a).

- (d) HDCD high-definition audio of 24-bit samples at 88.2 KHz

  Same as (a).

v.4.0.0 – 22/March/2016

Practical Exercises in Computer Networks and Distributed Systems

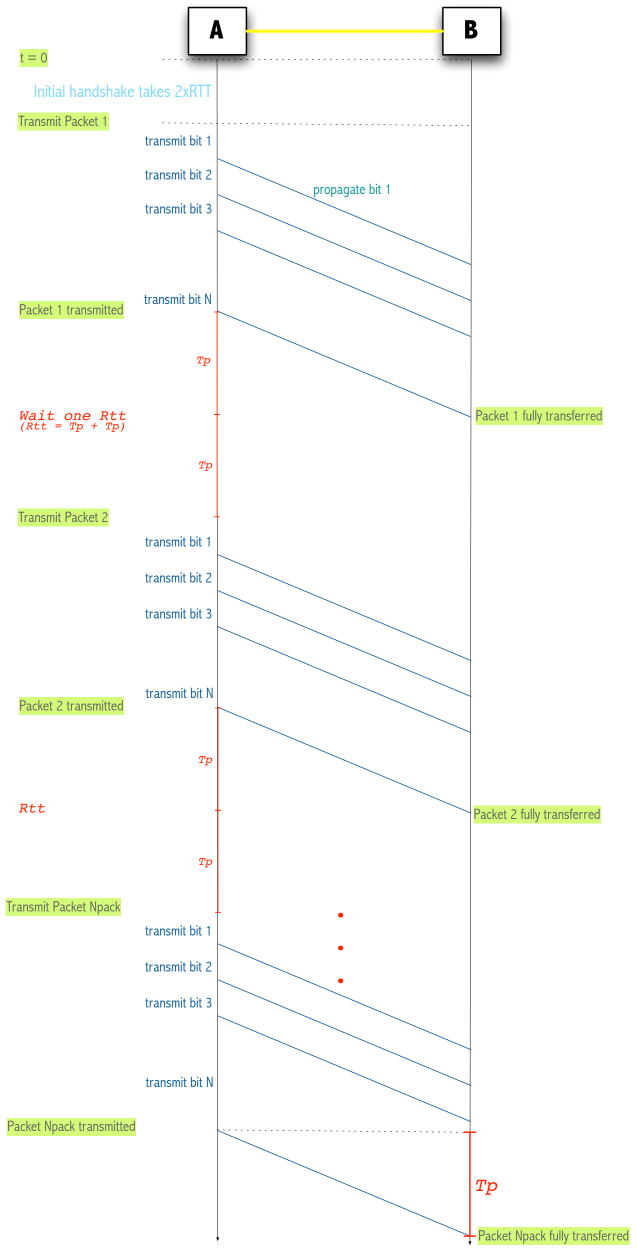
Complementary Notes to Network Performance

Ex. 4.b, solution 2

The adjoining figure aims to justify an alternative solution to exercise 4 that is more intuitive; in this case, to compute the total time it takes to transfer the 1.5MB we focus on the transmitting host A timeline. As before, we first obtain the formula that represents the total transfer time according to the diagram and, then calculate each of its components.

T_{total} =
Handshake +
$(T_{TransmPack} + Rtt)\cdot(Npack - 1)$ +
$T_{TransmPack}$ +
T_{prop}

WWW



(V.0.3, 27/3/2017)

Courses on Computer Networks and Distributed Systems

Complementary note on The Delay x Bandwidth product

Think of your uploading a large file to your Dropbox. Your host computer -the sender- might indicate its intention of starting the *upload operation* by sending a string of bits which meaning would be "UPLOAD REQUEST". The time it takes for the "UPLOAD REQUEST" to arrive at the receiver is roughly Rtt/2. The receiver, shortly after receiving the "UPLOAD REQUEST", would respond by sending an "UPLOAD ACCEPTED" message, which would also take Rtt/2 seconds to arrive at the sender. The sender had to wait a full Rtt to become convinced that the "UPLOAD REQUEST" it sent, was accepted. It sent the *few bits* that comprise the "UPLOAD REQUEST" message, and then it waited for the response for Rtt seconds.

If we measured the throughput in bits per second (bps), we would observe the result is shockingly low, and the reason is that the network, in fact could absorb many, many more bps than we offered –the bits sent in the text "UPLOAD REQUEST". You would argue that, when sending such short messages, the throughput attainable can not be high since their short nature precludes a high network utilization up front.

Diagram in Fig. 1 summarizes this idea, that it takes a full Rtt to receive the response after the request is sent and, furthermore, that after sending the REQUEST, the sender is sending no more bits until after it receives the REPLY. We aim to establish that continuous sending is a caveat for attaining any high level of throughput –at least, continual transmission would be required.

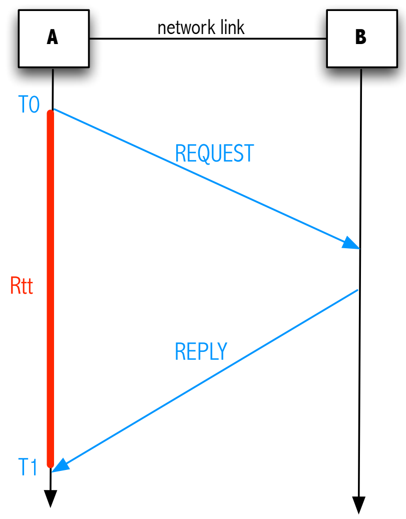


Fig. 1. Stop-and-wait: Sender sends *a few bits* (the REQUEST) and waits for the REPLY before proceeding with sending the file blocks

A positive response from host B (The receiver) means that host A is allowed to upload the file, thus, it would start sending the first file blocks shortly after the reply is received. However, why not start sending the first blocks right after the request? If the reply from B were negative, it would discard them altogether; however, in case the reply were positive, host B would have received the first blocks amidst a time that would otherwise have been wasted. It looks as though a performance increase could be accomplished if sender and receiver were able to function with continuous –or at least, continual- sending instead of sending and stopping for the reply. The transmission diagram in fig. 2 will help us summarize the idea of continual transmission *vs.* stop-and-wait. Continuous transmission allows the sender to transmit at B bps without ever stopping transmission, whereas continual transmission also allows B bps transmission bandwidth but it prescribes that transmission can be stopped for short periods of time, according to the needs of the protocol governing the transfer between the two hosts. The latter strategy results more realistic than the former.

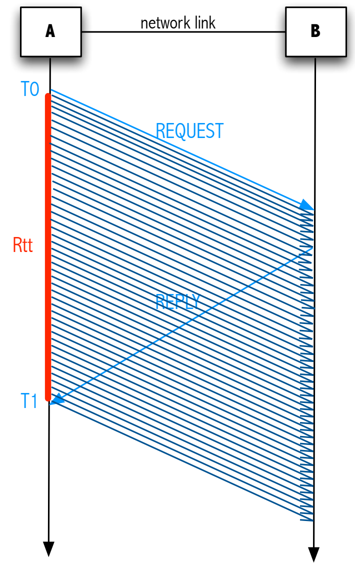


Fig. 2. Sender A sends at B bps for the whole duration of the Rtt, then it achieves the maximum performance attainable (Continuous transmission)

In interpreting fig. 2, the essential point consists of assuming that Rtt is fixed, it can obviously change with time, however, for the purposes of grasping the utility of the Bandwidth x Delay product, we assume the Rtt is fixed for the time length of our experiments, then, the faster the transmitter, the better: the tighter the lines in fig. 2, the better, because we are transmitting more bits per unit time. The Rtt is fixed; then, we should transmit as fast as we can and without any pauses whatsoever. This continual transmission requires our smart designing the protocol between A and B, a protocol that attains that A will not be obligated to ever stop transmitting. That's the challenge. Could we somehow quantify how smart should our protocol design be? In other words, when designing the peer-to-peer interface operating between sender and receiver, what quantitative measure could become our benchmark?

Assume Bandwidth is B bps, then, while waiting for the REPLY, the sender keeps sending at B bits-per-second (bps) for Rtt seconds. Finally, when the REPLY arrives and we ask how many bits got sent meanwhile, the answer comes as the following expression, known as the B x Rtt product:

$$\underline{\text{B x Rtt}} = \text{B bits/}\cancel{\text{sec}}\text{ x Rtt }\cancel{\text{sec}} = \text{B x Rtt bits}$$

Under normal circumstances, conventional network designs will be such that Rtt = 2 x $D_{\text{one-way-delay}}$, in other words, the round trip time will be equal to the sum of the delay in the forward direction and the delay in the backward direction. It's customary to refer to the bandwidth x delay product simply as 2BD:

$$\text{Bandwidth x Delay} = \text{Bandwidth} \cdot 2 \text{ x } D_{\text{one-way-delay}} = 2BD$$

Do we wish to have networks with large Rtts? Since having a large Rtt means we keep transmitting for longer times, that could give us the wrong impression that large Rtts are something we must seek. No, a large Rtt means some applications will not function properly, for example, digital telephony which prescribes a max Rtt of 200ms, beyond which, the conversation parties will begin to lose the perception of conversation and, somehow they will be forced to enter a conversation mode that might resemble that of a pair of walkie-talkies. In summary, large delays are to be avoided altogether, however, if a network presents a high Rtt, we seek to take advantage of it by having our protocol keep sending while it's waiting for any response. In the preceding diagram, we are taking advantage of the Rtt by designing a peer-to-peer interface (A protocol) that properly handles the aforementioned, continual transmission.

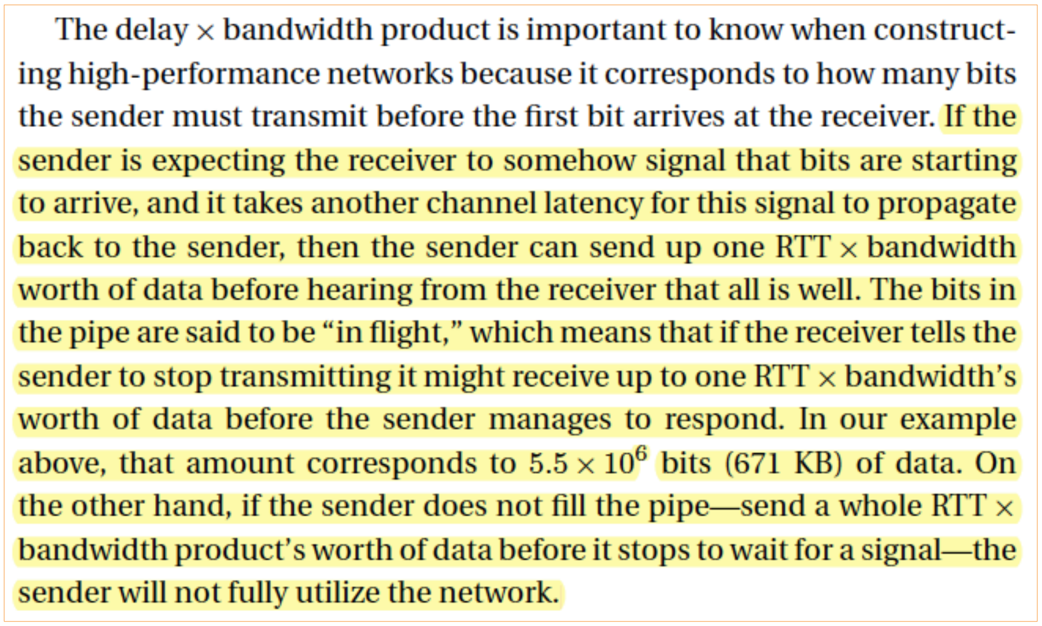
In designing a protocol that regulates data transfer between a sender and a receiver, the 2BD product establishes a maximum to the amount of data the sender can inject within a single Rtt. 2BD becomes our benchmark when establishing whether our protocol is performing well. The opportunity comes from large 2BD products, however large is B or D: if 2BD is large, then, a well-designed protocol can take advantage of it. Somehow, fig.1 and fig. 2 are extreme cases; in fig. 1 the sender instantaneously transmits a packet and waits a full Rtt for the indication from the receiver that it successfully received it, then, it continues to transmit the next packet and waits Rtt seconds again for the response. A protocol like this one is known as stop-and-wait. Right on the other side of the spectrum is the protocol in fig. 2, which keeps transmitting while waiting for the response, and never stops transmitting. This protocol at all times has 2BD bits in flight, which would maximize its throughput. It's difficult to design such a protocol and, since there exist other factors that affect the interaction between sender and receiver and which would cause the throughput to decrease, such as: bit-errors, packet-errors, packet-delay, packet loss, etc.

In summary, networks that have a large 2BD present a challenge to the designers: to take advantage of it by implementing a protocol that transmits continuously if possible, otherwise it transmits continually with as few as possible pauses between periods when sending and periods when not sending and guaranteeing those pauses to be as short as possible.

Questions and exercises.

1. Design an experiment for estimating the 2BD product of two Internet hosts which interconnection path contains 5 IP routers:
    a. All routers transmit at the same speed of B bps
    b. There's one router whose IP interface uses a speed of B/10 bps
2. The end-to-end bandwidth undergone by an Internet host when transmitting to another host is about 0,65Mbps, if the Rtt is 1µs, explain whether or not it would be convenient to write a fully pipelined transfer protocol or remain with the stop-and-wait version.
3.

Translation of paragraph in P&D pg. 49



The delay × bandwidth product is important to know when constructing high-performance networks because it corresponds to how many bits the sender must transmit before the first bit arrives at the receiver. If the sender is expecting the receiver to somehow signal that bits are starting to arrive, and it takes another channel latency for this signal to propagate back to the sender, then the sender can send up one RTT × bandwidth worth of data before hearing from the receiver that all is well. The bits in the pipe are said to be "in flight," which means that if the receiver tells the sender to stop transmitting it might receive up to one RTT × bandwidth's worth of data before the sender manages to respond. In our example above, that amount corresponds to 5.5×10^6 bits (671 KB) of data. On the other hand, if the sender does not fill the pipe—send a whole RTT × bandwidth product's worth of data before it stops to wait for a signal—the sender will not fully utilize the network.

Si el transmisor espera a que el receptor le indique que los bits transmitidos comienzan a llegar y, esta indicación consume otra latencia de canal (El tiempo de retorno de B-> A) en propagarse hasta el receptor, entonces, el transmisor puede enviar un máximo de B x Rtt antes de apercibirse de que todo va bien (Que los bits enviados han comenzado a llegar al receptor). Los bits que se encuentran en "el tubo" decimos que están "en vuelo", lo que significa que, si el receptor le indica al transmisor que deje de transmitir, podría recibir de éste hasta B x Rtt bits antes de que el transmisor responda consecuentemente…