Lab

4

# Bandwidth Estimation

## Purpose of this lab:

In this lab you will build a tool for estimating the available bandwidth of a network element. The network element consists of a black box with unknown properties. You will use probe traffic consisting of a sequence of packets to infer the properties of the network.

## Software Tools:

- The programming for this lab is done in Java.

## What to turn in:

- Turn in a report with your answers to the questions in this lab, including the plots, and copies of all your Java code.

**Note:** Part 3 is optional for max. 20 marks extra credit.

Version 1c (April 1, 2013)

## Table of Content



## Preparing for Lab 4

Review the concepts from previous labs, such as traffic generators, adding time-stamps to traffic, and saving data to files.

## Comments

- **Quality of plots in lab report:** This lab asks you to produce plots for a lab report. It is important that the graphs are of high quality. All plots must be properly labeled. This includes that the units on the axes of all graphs are included, and that each plot has a header line that describes the content of the graph.

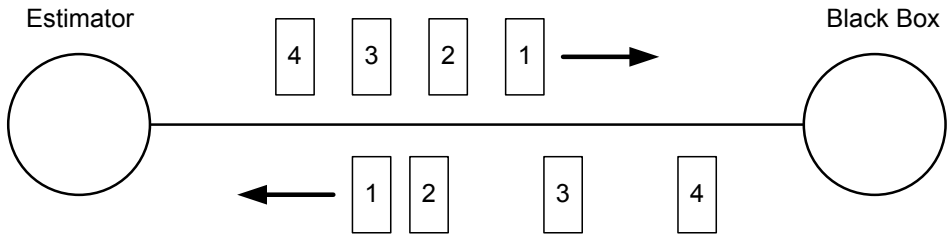
## Overview

The objective of this lab is to infer the available bandwidth in a network, where the network is represented by a "Black Box" component, from measurements of network probes. The Black Box is given to you.
It delays traffic by a constant amount T μsec, and transmits traffic at a long-term rate R Mbps, allowing transmission bursts of up to size b Bits. This results in an (exact) service curve:

$$S(t) = \begin{cases} b + R(t-T)\ , & t > T \\ 0\ , & t \le T \end{cases}$$

Note: In a packet-level implementation, the burst size b must be large enough to accommodate the largest packet size $L_{max}$, otherwise, a large packet may never be allowed to depart from the Black Box.

However, the values of b, R, and T are not known. The objective is to estimate the values of these parameters.




Your task is to build a software tool, called the "Estimator". The Estimator is a Java program which sends sequences of probe packets to the Black Box. The probe packets are returned by the Black Box to the Estimator. The Estimator creates timestamps for each packet sent to the Black Box and for each packet returning from the Black Box. Using these timestamps, the Estimator computes the parameters of the Black Box.

Probe packets are sent by the Estimator as sequences of packets, with equidistant gaps between the packets of a sequence. A single sequence of probe packets (of a given length) is called a *packet train*.
The **goal** is to estimate the parameters of the Black Box subject to the following **secondary goals**:

(1) The maximum rate at which a packet train is sent from the Estimator to the Black Box should not be larger than necessary;
(2) The total number of packet trains should be as small as possible;
(3) The length of packet trains should be as small as possible.

The remaining parts of the lab ask you to construct the Estimator and perform measurement experiments, where you estimate the parameters of BlackBox.

You will be working with three types of Black Boxes:

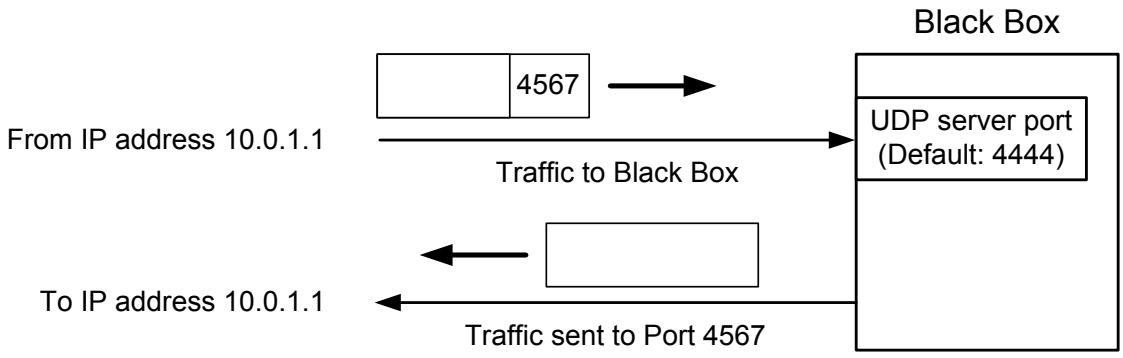
- For debugging and testing, you are given the source code of the Black Box.
- For evaluation, you are given compiled versions of the Black Box, with fixed but unknown parameters for b, R, and T.

- For a real-world experiment, we have placed Black Boxes in locations on the Internet. The locations and the addresses are available on the course website. For these Black Boxes, the parameters are set to ($b=L_{max}$, $R=\infty$, $T = 0$). With these settings, you will estimate the available bandwidth of the network path in the Internet between the Estimator and the Black Box.
- **In the provided implementation, the maximum packet size is set to 1480 Byte. So, we get $L_{max}$ =11840 bits.**

## BlackBox

The source code of BlackBox class and related classes are provided to you. The BlackBox receives UDP packets on a UDP (server) port. When a packet is received on the server port, the packet is processed according to the specified service curve, and then returned to the sender of the packet.

Important: The payload of each packet sent to BlackBox must have a number (e.g., 4567) as the first 2 bytes. BlackBox uses this number as the destination port number when returning a packet to the sender.




The UDP server port number is set from the command line. Starting BlackBox with

```
%java BlackBox  6543
```

creates an instance of the BlackBox with UDP server port number 6543. If no argument is given, the default port number 4444 is selected.

When no parameter is given, BlackBox uses as default port number 4444.

The parameters of the service curve implemented by BlackBox are hard coded in variables TConstant, RConstant and bConstant in BlackBox.java:

| | | |
|---|---|---|
| TConstant | – | delay parameter (in microseconds), |
| RConstant | – | rate parameter (in Mbps), |
| bConstant | – | burst parameter (in Bits). |

## Reading and writing numbers in the payload of a packet

Below we provide sample code for (1) setting bytes in the payload of a packet to a number, and for (2) extracting bytes in the payload of a packet and converting them to a number.

### Writing a port number into a packet

The following method converts an integer to a byte array.

```
/**
 * Converts an integer to a byte array.
 * @param       value   an integer
 * @return      a byte array representing the integer
 */
public static byte[] toByteArray(int value)
{
       byte[] Result = new byte[4];
       Result[3] = (byte) ((value >>> (8*0)) & 0xFF);
       Result[2] = (byte) ((value >>> (8*1)) & 0xFF);
       Result[1] = (byte) ((value >>> (8*2)) & 0xFF);
       Result[0] = (byte) ((value >>> (8*3)) & 0xFF);
       return Result;
}
```

This produces a byte array of with 4 bytes. If the value passed to the method is a short (a 16 bit integer used for port numbers) its value will be stored in the last 2 bytes of the array (with the first two bytes set to zero).

To add the port number `returnPort` to the first two bytes of the array `buf`, the following code can be used:

```
byte[] buf  = new byte[10];

// put returnPort in first 2 bytes of buf
System.arraycopy(toByteArray(returnPort),2,buf,0,2);
```

The first line creates a byte array of 10 bytes. The second line writes the value of returnPort into the first 2 bytes of `buf`. The method `toByteArray(returnPort)` returns a `byte` array (with length 4 bytes) containing the value of `returnPort`. The call `System.arrayCopy` copies the third and fourth bytes into the byte array `buf` (that is, in `buf[0]` and buf[1]).

Note: Calling

`System.arraycopy(source, srcPosition, dest, destPosition, numElements)`

copies `numElements` elements from the array `source`, beginning with the element at `srcPosition`, to the array destination starting at `destPosition`.

**Extracting the port number from an incoming packet**

The following method converts a byte array to an integer:

```
/**
 * Converts a byte array to an integer.
 * @param value            a byte array
 * @param start            start position in the byte array
 * @param length           number of bytes to consider
 * @return                 the integer value
 */
public static int fromByteArray(byte [] value, int start, int length)
{
       int Return = 0;
       for (int i=start; i< start+length; i++)
       {
              Return = (Return << 8) + (value[i] & 0xff);
       }
       return Return;
}
```

This method extracts an integer from a byte array value. Since an integer is represented by 32 bits =4 bytes, the value of length should not exceed four.

To extract a port number from the first two bytes of from the payload of a UDP packet, the following code can be used:

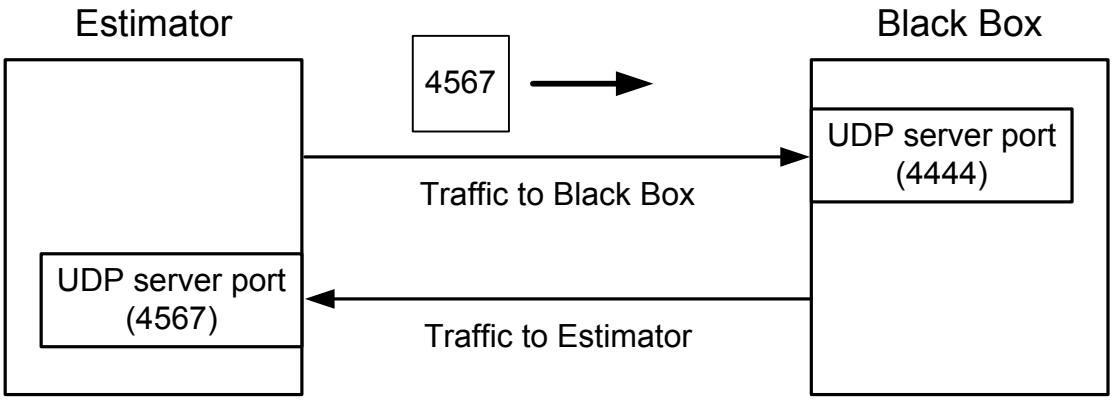
```
int portNumber = fromByteArray(packet.getData(), 0, 2);
```

## Part 1. Generating and Time-stamping Packet Trains

In this part of the lab, you create components of the Estimator for

- Sending packets to and receive packets from BlackBox;
- Sending and receiving a packet train with sequence numbers;
- Managing timestamps of packets sent to and received from the BlackBox;
- Creating and sending packet trains at a given rate;

The Estimator combines aspects of the Traffic Generator and Traffic Sink from previous labs. The Estimator sends packets to a UDP (server) port of the Black Box, and receives packets from the Black Box on a UDP (server) port.




The UDP server ports of the Estimator and BlackBox are created when the applications are started. If the BlackBox is running remotely, the number of the UDP server port must be published together with the IP address. An issue with a remote (or shared used) Black Box is that BlackBox does not have a priori knowledge of the server port number of the Estimator. Thus, we add the requirement that the port number of the UDP server port of the Estimator is included in each packet sent from the Estimator to BlackBox.

The specific requirement is that the first 2 bytes of the payload of each probe packet must contain the port number of the UDP server port at the Estimator.

The following exercises breaks up the task of creating the Estimator application into smaller pieces. You may decide to combine the exercises, or build the pieces in a different order.

### Exercise 1.1 Exchanging Traffic with the BlackBox

Write a program that sends UDP datagrams to a running instance of the BlackBox, and accepts the returning packets from BlackBox.

- Start an instance of BlackBox from the command line. For example the command

```
java BlackBox 6543
```

creates an instance of BlackBox with UDP server port number 6543. If no argument is given, the default port number 4444 is selected.

By default, the parameters of BlackBox are b= b=$L_{max}$, R=∞, T = 0. If you want to use different parameters modify the constants *bConstant*, *RConstant*, and *TConstant* in the BlackBox.java and recompile BlackBox.java.

- Your implementation must be able to send UDP datagrams to a BlackBox instance for a given IP address and UDP port on the Internet. This can be achieved by providing the IP address and port number as arguments when you start the Estimator (e.g., java Estimator 10.0.1.2 4444), or by providing data interactively by prompting for user input.

- For the Estimator, you may use parts of the source code from the Traffic Generator and Traffic Sink from previous labs, as well as the provided source code of the Black Box.

- For writing the UDP server port number in the first 2 bytes of a packet you must turn an integer into a byte array, and copy the byte array in the payload of the packet. Source code of a function toByteArray which turns an integer into a 4-byte long byte array is provided to you.

- Note: When BlackBox cannot read the port number in the first byte of an incoming datagram, it will send the datagram to port number 4445.

At the end of this exercise, your Estimator should be able to send several (more than one) probe packets to BlackBox, and receive and process the incoming probe packets that are returned by BlackBox.

## Exercise 1.2 Adding Sequence Numbers to Probe Packets

The next step is to add sequence numbers to probe packets sent by the Estimator. You must be able to recognize the sequence number of an incoming probe packet that has been sent by BlackBox. The sequence numbers are used to match outgoing and incoming probe packets.

- For a sequence of probe packets (packet train), the sequence number of a packet is its position in the sequence. The first probe packet has sequence number “1”, the second has sequence number “2”, and so on.

- The sequence number of a packet must be converted to a byte array and then added to the probe packet. It is sensible to write the sequence number in the bytes following the port number (see previous exercise).

- When a packet is returned from BlackBox, you must be able to read and process (e.g., display) the sequence number of an incoming packet.

- Test the traffic sink with the traffic generator from Part 2 of Lab 2a.

## Exercise 1.3 Record Timestamps

The next step is to record timestamps for each probe packet, together with the sequence. Two timestamps must be recorded for each probe packet:

(1) Send Timestamp: The Send Timestamp is recorded when a probe packet is transmitted.

(2) Receive Timestamp: The Receive Timestamp is recorded when a probe packet is received from BlackBox.

The unit of the timestamps should be in microseconds (μsec). Timestamps must be normalized to the Send Timestamp of the first probe packet, whose timestamp is set to zero.

You need to create data structures that records the sequence numbers of packets together with the timestamps, e.g.,

| SeqNo | Send Timestamp (in μsec) | Receive Timestamp (in μsec) |
|---|---|---|
| 1 | 0 | 15 |
| 2 | 100 | 122 |
| 3 | 200 | 212 |

Note: In Exercise 2.5, you need to save the timestamps to a file, and create plots from the saved data.

## Exercise 1.4 Packet Trains with Given Parameters

The next extension enables the Estimator to transmit a packet train of probe packets for given parameters. The parameters of the packet train are

- N: number of packets of the train;
- L: packet size of the train (in Byte);
- r: average bit rate of the train (in kbps).

The time interval between the transmission of two packets is constant. The length of the interval can be computed with parameters L and r.

Your implementation of the Estimator must be able to transmit multiple packet trains with different values for N, L, r. (In other words, the parameters cannot be constants in the source code).

## Exercise 1.5 Evaluation

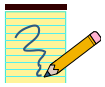
Run an experiment where you send traffic to BlackBox and take measurements of the results.

- Create a BlackBox with parameters $b=L_{max}$, $R=\infty$, T = 1000. This is done by modifying the parameter TConstant in BlackBox.java, re-compiling, and starting an instance of BlackBox.

- Use your implementation of Estimator to measure the timestamps of three packet trains with the following parameters and save the timestamps to a file:

    1. Packet train: N=100, L=400, r=10;
    2. Packet train: N=100, L=400, r=1000;
    3. Packet train: N=100, L=400, r=10000.

- From the saved timestamps, prepare plots (one for each packet train) where the sequence number is on the x-axis, and the value of the timestamps is on the y-axis. Each plot has two curves: (1) for the Send Timestamps and (2) for the Receive Timestamps.



**Lab Report:**

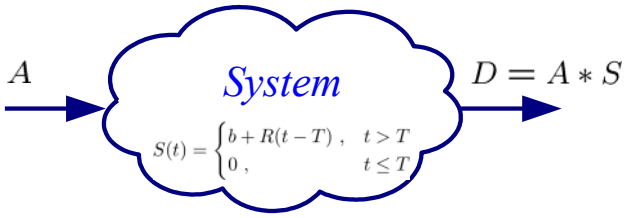
Provide the plots and a discussions of the graphs generated in Exercise 2.5. Include the source code for the Estimator, that was used for Exercise 2.5.

## Part 2. Bandwidth Estimation of Black Boxes

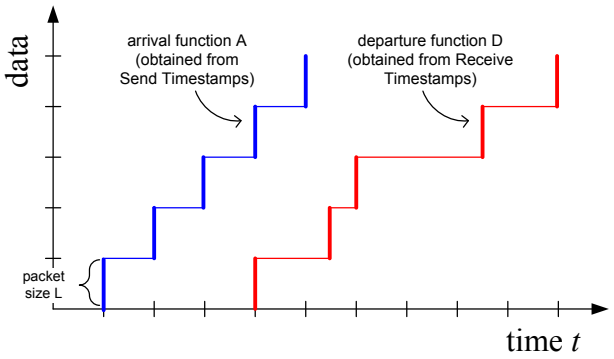
In this part of the lab, you work with the Estimator from Part 1 for an estimation of the available bandwidth for a number of (provided) BlackBoxes with unknown parameters b, R, and T.

Your objective is to find the value of the unknown parameters, from the saved timestamps of a series of packet trains.

If we view the BlackBox as a min-plus linear system, we can use the deterministic network calculus to compute the parameters of the service curve. Here, we view the BlackBox as a system, where the arrivals consist of the probe packets of a single packet train sent by the Estimator to BlackBox, and the departures consist of the probe packets that BlackBox returns to the Estimator.



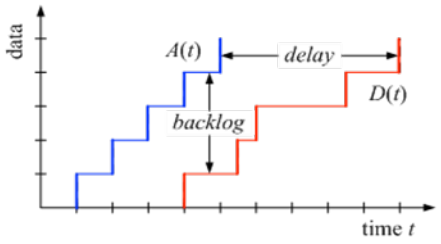

Using the timestamps of a packet train saved by the Estimator and the packet sizes, we can construct the arrival function A and the departure function D of the packet train, which may look as follows:




Then the problem of estimating the unknown parameters consists of the problem of solving the equation D (t) = A*S (t) for S. For packet trains which are sent at a constant rate, that is, A(t) = r t, we can make the following derivation.

- **Backlog:** $B(t) = A(t) - D(t)$
- **Maximum backlog:**

$$B_{max} = \sup_t \{A(t) - D(t)\}.$$




- **If** $A(t) = rt$ **n write this as:**

$$\begin{aligned} B_{max}(r) &= \sup_t \{rt - \inf_\tau \{r\tau + S(t-\tau)\}\} \\ &= \sup_t \{\sup_\tau \{r(t-\tau) - S(t-\tau)\}\} \\ &= \sup_t \{rt - S(t)\} \\ &= \mathcal{L}_S(r) \end{aligned}$$

- **Inverse transform:** If $S$ is convex we have

$$S(t) = \mathcal{L}(\mathcal{L}_S)(t) = \mathcal{L}_{B_{max}}(t) = \sup_r \{rt - B_{max}(r)\}$$

This suggests a probing methodology as follows:

**Initialize:** Set $\mathcal{R} = \emptyset$ and $S(t) = 0$.

1. Pick a new rate $r$, and add $r$ to the rate set $\mathcal{R}$, i.e., $\mathcal{R} = \mathcal{R} \cup \{r\}$.
2. Transmit a packet train at rate $r$.
3. From the timestamps, create the arrival function A and departure function D.
4. From A and D, compute $B_{max}(r)$.
5. Re-compute the service curve estimate:
$$S(t) = \max_{r \in \mathcal{R}} (rt - B_{max}(r))$$
6. If the service curve estimate has improved (is larger than the previous estimate) jump to Step 1.
7. Return the function $S(t)$.

Using this algorithms, you can obtain the values of T, b, and R from the service curve $S(t)$.

## Exercise 2.1 Implement and test the probing methodology

Implement the above probing methodology as an addition to the Estimator from Part 1. Note that the description above requires you to make certain choices:

- Choice of the packet size L and number of packets in a packet train N. (You can start with L=400 Byte and N=100.)

- Selection of the probing rates $r$.

Evaluate your implementation by performing a bandwidth estimation of three instances of BlackBox.

- Create a set of BlackBox instances with the following parameter settings:

  **BlackBox 1**: b=$L_{max}$, R=100, T = 10000;

  **BlackBox 2:** b=10000, R = 100, T = 100,000;

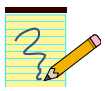
  **BlackBox 3:** b=$L_{max}$, R = 1000, T = 20000.

- Use your implementation of the probing methodology to determine the service curves for a BlackBox instance.
- Create plots of the service curves, and compare them to the ideal service curve.
- Repeat the experiments with the same parameters. Do you get the same results?
- Repeat the estimation methodology of the three BlackBox instances using longer packet trains (double N), and a larger packet size (e.g., double the packet size). Include the results of the service curve into the graphs from the previous step.

## Exercise 2.2 Evaluation of BlackBox with unknown parameters

From the course webpage, download compiled versions of the three BlackBox instances with fixed but unknown parameters for b, R, and T.

- There are three BlackBox instances to download. Each Black Box has the default UDP server port number 4444.
- Use your version of the Estimate with the probing methodology to determine the service curves for each BlackBox instance.
- Create plots of the service curves.
- Provide estimates of the parameter values for b, R, and T.
- Repeat the experiments with the same parameters. Do you get the same results?



**Lab Report:**

- Provide a description how you implemented the probing methodology as an addition to the Estimator:
  - Discuss the your algorithm for selecting the rates.

- Include the source code for the Estimator, that was used for Exercises 2.1 and 2.2.

- Present and discuss the results of your measurement experiments. Include the plots and a description of the plots. Express your degree of confidence in these estimates.

- For Experiment 2.1, discuss when your methodology was able to find the correct parameters and when it had difficulties finding the correct parameters. Also, describe the impact of increasing the length of the packet trains and the size of the probe packets on the outcome of the measurements.

- For Experiment 2.2, provide a discussion of your estimates of the values for b, R, and T.

- Report your findings when you repeated the measurement experiments.

## Part 3. (optional: max. 20 marks) Bandwidth Estimation of the Internet

In this part of the lab, you use the probing methodology implemented in the Estimator to estimate the available bandwidth of remotely running instances of the Black Box.

The location of the Black Boxes, as well as their IP addresses and the UDP server port numbers will be available on the course website. For the remote Black Boxes, the parameters are set to $b=L_{max}$, $R=\infty$, $T = 0$. In other words, the Black Box does not impose delay or a rate limitation and its service curve is the burst function $\delta(t)$. On the other hand, the probing traffic will traverse networks that impose rate limitations as well as delays.

- Use the probing methodology implemented in your Estimator from Part 2 to obtain estimates of the service curve.
- The conditions on the network path from the Estimator and Black Box are variable due to the randomness of the amount of traffic from other traffic sources. You can get a handle on the randomness by repeating the measurements many times. Provide a large (min. 20) repetitions and obtain the service curves.
- Generate plots of the service curves.

Note:

- When too many groups are running the experiments at the same time, the Black Box may become a bottleneck and the outcome may change accordingly.
- The probing of the remote BlackBox will not work from the machines in GB243, since the firewall of the lab does not let UDP traffic pass. You should run the test using the UofT wireless network or your home network.

**Lab Report:**

- Present and discuss the results of your measurement experiments. Include the plots and a description of the plots. Express your degree of confidence in these estimates.