**Project 2: Batcher Banyan Network**

Sushmitha Alagesan (1223040819)  
Arizona State University  
IFT 510: Principle of Computer and Information Technology

Prof. Edmund Lorenzen  
April 29, 2022

**1.Batcher -Banyan network:**

Batcher-Banyan networks are a type of multistage interconnection network that is commonly found in multiprocessor interconnections and ATM switches.

A Banyan network could potentially transport data packets from any input to any output, but conflicts at any intermediate state would result in the loss of packets being carried through. To avoid packet loss in the delta network, we sort the data first before delivering it to the banyan(delta) network, which then discovers and routes each packet correctly.

**1. Develop the expected values of any tests conducted that demonstrate how your program   
functions.**

The working of the Batcher – Banyan network as implemented in the program has been illustrated below with a couple of examples:

Chart, diagram

Description automatically generated

**Logic:**

1.The number of inputs is obtained from the user and random numbers are generated for the given count

2. This input first passes through the batcher network, which arranges the data according to the banyan network's desired positions. Then the sorted data is displayed.

3. Determine each data's binary values . For each binary value, a column state is generated

4. Implement a switching pattern at each column state A|B|C|D to determine the route that each packet will follow, for example, if the packet is 4 ==> 0100, 4 will first travel to A5, as shown in the figure above, from A5 if the first value of binary is 0, go to B1, otherwise B5.

5. Similarly, we verify the binary value (0100 - 2nd column) from B1 and move to c1 if value is 0, otherwise C3 in this manner, here it moves to C3 due to 1, then from c3 to D3 then out 4 .We will be able to reach the destination output port appropriately once we have traversed all the switches.

6. considering high value 15(1111) all the columns choose to take the down path because of 1 . so the route will be A8 to B8 to C8 to D8 and finally out 15

**Output Result:**

**Graphical user interface, text, application

Description automatically generated**

**2. Provide a copy of your source code.**

**Language used : python**

**Source Code:**

# Author @ Sushmitha Alagesan Date: friday April 29

# random funtion in python is called for generating random numbers

import random

# If given n bits of binary numbers this function splits all the binary numbers in digit wise

# In this function the decimal digits are converted to binary by bin()

# this gives the binary equivalent string of a given integer

def deci\_to\_binary(num\_bits, number):

bin = ''

while True:

bin = str(number%2) + bin

number = number//2

if number == 0 or number == 1:

bin = str(number) + bin

break

if len(bin) == num\_bits:

return bin

elif len(bin) < num\_bits:

return '0'\*(num\_bits-len(bin)) + bin

else:

return 'overflow'

# we pass them to the dedicated slot in the banyan network

# The binary numbers are 0's and 1's ,if 0's then it is mapped to the top portion of routing network

# If 1's then it is mapped to the bottom portion of routing network

# For Batcher-Banyan network,Each bit is represented as each column ,

# here we give 4 bits which splits in to 4 columns A,B,C,D

def bat\_banyan(b, switch\_route, one\_bit):

print('-->', switch\_route if one\_bit == '0' else switch\_route[0] + str(int(switch\_route[1])+4), end=' ')

if b[0] == '0':

b = b[1:]

if switch\_route == 'B1':

return bat\_banyan(b, 'C1', one\_bit)

elif switch\_route == 'B2':

return bat\_banyan(b, 'C2', one\_bit)

elif switch\_route == 'B3':

return bat\_banyan(b, 'C1', one\_bit)

elif switch\_route == 'B4':

return bat\_banyan(b, 'C2', one\_bit)

elif switch\_route == 'C1':

return bat\_banyan(b, 'D1', one\_bit)

elif switch\_route == 'C2':

return bat\_banyan(b, 'D1', one\_bit)

elif switch\_route == 'C3':

return bat\_banyan(b, 'D3', one\_bit)

elif switch\_route == 'C4':

return bat\_banyan(b, 'D3', one\_bit)

else:

return '--> out' + str(1 + int(switch\_route[1])\*2 - (2 if one\_bit == '0' else -6))

else:

b = b[1:]

if switch\_route == 'B1':

return bat\_banyan(b, 'C3', one\_bit)

elif switch\_route == 'B2':

return bat\_banyan(b, 'C4', one\_bit)

elif switch\_route == 'B3':

return bat\_banyan(b, 'C3', one\_bit)

elif switch\_route == 'B4':

return bat\_banyan(b, 'C4', one\_bit)

elif switch\_route == 'C1':

return bat\_banyan(b, 'D2', one\_bit)

elif switch\_route == 'C2':

return bat\_banyan(b, 'D2', one\_bit)

elif switch\_route == 'C3':

return bat\_banyan(b, 'D4', one\_bit)

elif switch\_route == 'C4':

return bat\_banyan(b, 'D4', one\_bit)

else:

return '--> out' + str(1 + int(switch\_route[1])\*2 - (1 if one\_bit == '0' else -7))

network\_choise = int(input('PRESS 1, if complete Batcher-Banyan network is required\nPRESS 2, if only the Banyan Press is required 2\n Input number: '))

# Batcher-Banyan network is choosen

if network\_choise == 1:

n\_bit = 4

n\_range = 15

# Banyan press is choosen

elif network\_choise == 2:

n\_bit = 3

n\_range = 7

random\_arr = []

# The number is choosen from 1 to range+1 as the range starts from 0

while True:

print('\nEnter a number which ranges from 0 to', n\_range+1)

# the input number is obtained from the user

number = int(input('Number of random inputs: '))

# the number entered should be within the range specified

if n\_range+1 >= number >= 0:

break

# random numbers are generated for the given number of input counts.

# This is done by using import random libraries

# then the obtained random numbers are stored in the array random\_arr[]

for \_ in range(number):

random\_int = random.randint(0, n\_range)

while random\_int in random\_arr:

random\_int = random.randint(0, n\_range)

random\_arr.append(random\_int)

# the numbers in the random\_arrrand\_arr[] is then printed

print(random\_arr)

# In order to avoid packet loss or collision, the numbers in the array are sorted

random\_arr.sort()

print('After Sort...')

# the the numbers are printed after sorting

print(random\_arr)

# if batcher-Banyan network is choosed

# the for loop runs untill all the random inters in the array are routed to particular output

# the input numbers are converted to binary by calling deci\_to\_binary funtion

# routing is performed by calling banyan function

if network\_choise == 1:

for i in range(len(random\_arr)):

print(random\_arr[i], ' --> ', end=' ')

switch\_route = 'A' + (str(i+1) if i < 8 else str(i-7))

print(switch\_route, end='')

task = deci\_to\_binary(4, random\_arr[i])

if task[0] == '0' and int(switch\_route[1]) < 5:

print(bat\_banyan(task[1:], 'B' + switch\_route[1], '0'))

elif task[0] == '0' and int(switch\_route[1]) > 4:

print(bat\_banyan(task[1:], 'B' + str(int(switch\_route[1])-4), '0'))

elif task[0] == '1' and int(switch\_route[1]) < 5:

print(bat\_banyan(task[1:], 'B' + switch\_route[1], '1'))

elif task[0] == '1' and int(switch\_route[1]) > 4:

print(bat\_banyan(task[1:], 'B' + str(int(switch\_route[1])-4), '1'))

# if Banyan press is choosed

else:

for i in range(len(random\_arr)):

print(random\_arr[i], end=' ')

switch\_route = 'A' + (str(random\_arr[i] + 1) if random\_arr[i] < 8 else str(random\_arr[i] - 7))

print(bat\_banyan(deci\_to\_binary(3, random\_arr[i]), 'B' + (str(i+1) if i < 4 else str(i-3)), '0'))

# Author @ Sushmitha Alagesan Date: friday April 29

**4.Provide a detailed discussion of the results and the concepts you used in developing the   
program. Discussions should include analysis and a comparison of the results with the   
expected value from the theory.**

**Screenshot of the code with explanation:**

* A random function in Python is invoked in this block of code to generate random integers.
* Bin() converts the decimal number created at random to binary, yielding the binary corresponding string.
* This function separates all binary integers digit by digit if provided n bits of binary numbers.

**Graphical user interface, text, application

Description automatically generated**

* The digits are sent to a specific slot in the banyan network in this block of code.
* The binary numbers are 0's and 1's; if 0's, the top portion of the routing network is mapped to it.
* If it's a 1, it'll be assigned to the bottom of the routing network.
* Each bit is represented as each column in the Batcher-Banyan network, and here we have four bits that are divided into four columns: A, B, C, and D.
* The check for batcher banyan 16x16 or 8x8 which is banyan press is also done here

**Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated**

* In this section of code, the number is chosen from 1 to range+1, as the range begins at 0; the user's input number is retrieved; the number entered must be within the range indicated; random numbers are generated for the given number of input counts.
* This is accomplished by importing random libraries, and then storing the resulting random numbers in the array random arr[].
* The random numbers in the array are then printed.
* The integers in the array are sorted to avoid packet loss or collision.
* If the batcher-Banyan network is selected, the for loop runs until all of the random integers in the array are routed to a certain output.
* The input numbers are converted to binary using the deci\_to\_binary function, and the routing is done using the banyan function.

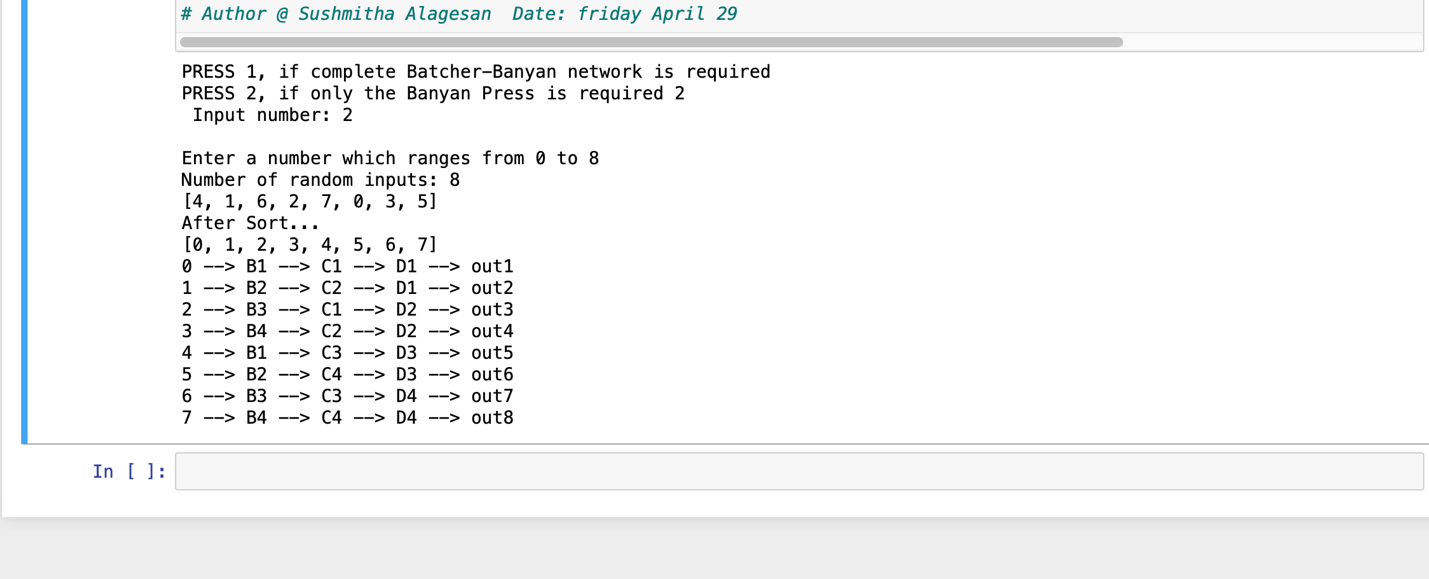
**Graphical user interface, text, application

Description automatically generatedText

Description automatically generated**

**Output Screenshot:**





Question 1: For a few random numbers, we manually generated a batcher banyan network, which we compared to the results of the executed program and found to be quite similar, with no inconsistencies.

**3.Provide a screenshot (or series of screenshots within your video video) demonstrating   
how your simulation works for all inputs and outputs – 0-7.**

**Input:0**

**Graphical user interface, text, application, email

Description automatically generated**

**Input:1**

**Graphical user interface, text, application

Description automatically generated**

**Input:2**

**Graphical user interface, text, application, email

Description automatically generated**

**Input:3**

**Graphical user interface, text, application

Description automatically generated**

**Input:4**

**Graphical user interface, text

Description automatically generated**

**Input:5**

**Graphical user interface, text

Description automatically generated**

**Input:6**

**Table

Description automatically generated with medium confidence**

**Input:7**

**Graphical user interface, text, table

Description automatically generated**

**3. Ensure you separate the   
Batcher sorter from the Banyan network and show how each work**

The batcher sorter will take the user's input value and produce random numbers depending on it, before sorting the numbers upwards.

The following is the batcher sorter's source code and how it works:

**Source code:**

# Author @ Sushmitha Alagesan Date: friday April 29

import random

network\_choise = int(input('PRESS 1, if complete Batcher-Banyan network is required\nPRESS 2, if only the Banyan Press is required 2\n Input number: '))

# Batcher-Banyan network is choosen

if network\_choise == 1:

n\_bit = 4

n\_range = 15

# Banyan press is choosen

elif network\_choise == 2:

n\_bit = 3

n\_range = 7

random\_arr = []

# The number is choosen from 1 to range+1 as the range starts from 0

while True:

print('\nEnter a number which ranges from 0 to', n\_range+1)

# the input number is obtained from the user

number = int(input('Number of random inputs: '))

# the number entered should be within the range specified

if n\_range+1 >= number >= 0:

break

# random numbers are generated for the given number of input counts.

# This is done by using import random libraries

# then the obtained random numbers are stored in the array random\_arr[]

for \_ in range(number):

random\_int = random.randint(0, n\_range)

while random\_int in random\_arr:

random\_int = random.randint(0, n\_range)

random\_arr.append(random\_int)

# the numbers in the random\_arr is then printed

print(random\_arr)

# In order to avoid packet loss or collision, the numbers in the array are sorted

random\_arr.sort()

print('After Sort...')

# the the numbers are printed after sorting

print(random\_arr)

# Author @ Sushmitha Alagesan Date: friday April 29

**Screenshot:**

**Graphical user interface, text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated**

**Advantages of this code**

1. It accepts up to 16 numbers from the user in a dynamic manner.

2. Accepts any input  >= 0 and <= 15

**Disadvantages of this code**

1. We are unable to go higher in order at this time; 32 bit or higher implementations are not supported.

**5. Link of the video file:**

<https://pro.panopto.com/Panopto/Pages/Viewer.aspx?tid=25583167-a590-46b1-8566-ae87005a86ff>

Note: mp4 file has also been attached along with the word file