POLITECNICO DI TORINO

Exercises on Switching Architectures, Data Centers and Fast Packet Processing

Class of *Switching Technologies for Data Centers*

Academic year 2021/22

Paolo Giaccone

Version: September 28, 2021

Contents

Chapter 1

Interconnection networks

1.1 Clos networks

Exercise 1
Design a Clos network, strictly non blocking, of size 100×100*, using modules* 10×10*. Compute the final complexity in function of the complexity* $\mathcal{C}(10)$ *of the* 10×10 *module.*

Solution:

$$N = 100$$

$$p = 10 = \sqrt{N} \Rightarrow q = 10$$

$$l = 2p - 1 = 19$$

The resulting network is shown in figure 1.1.

To build a 10×19 module, it is possible to use two modules 10×10 in parallel, of which the first 19 outputs are connected to the following 19 modules of the second stage, as shown in figure 1.2. The last output of the second 10×10 module in parallel will be idle, but it will be included in the computation of the total complexity of the network.

$$\mathcal{C}_{SNB}(100) = 10 \times 2\mathcal{C}(10) + 19\mathcal{C}(10) + 10 \times 2\mathcal{C}(10) = 59\mathcal{C}(10)$$

★★★

Exercise 2
Design a Clos network, strictly non blocking, 1000×1000*, using only* 10×10 *modules. Compute the final complexity in function of* $\mathcal{C}(10)$*.*

Solution:

$$N = 1000$$

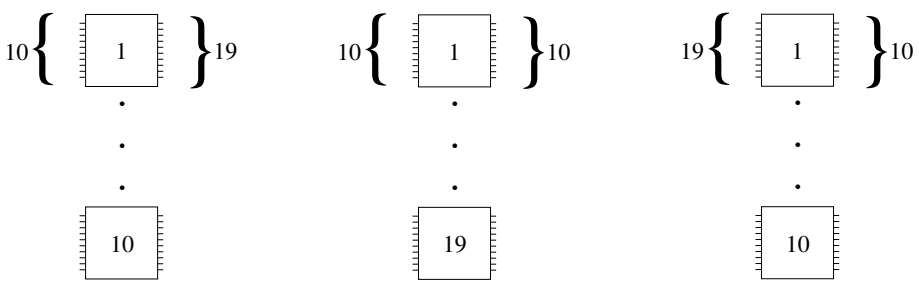


Figure 1.1: Strictly non blocking Clos network 100×100

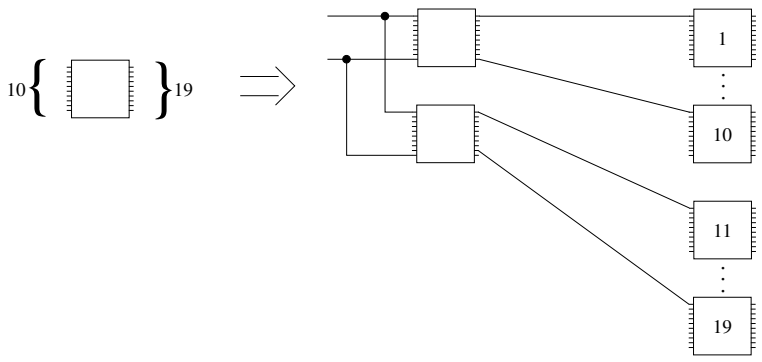


Figure 1.2: Architecture of a 10×19 module

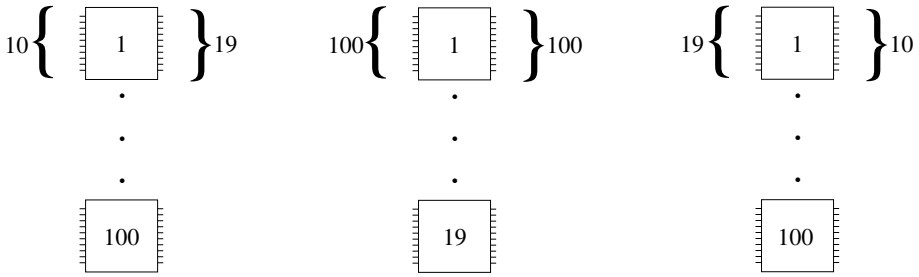


Figure 1.3: Strictly non blocking Clos network 1000×1000

$$p = 10$$
$$q = \frac{N}{p} = 100$$
$$l = 2p - 1 = 19$$

The resulting network is shown in figure 1.3.

$$\mathcal{C}_{SNB}(1000) = 100 \times 2\mathcal{C}(10) + 19\mathcal{C}_{SNB}(100) + 100 \times 2\mathcal{C}(10)$$
$$\mathcal{C}_{SNB}(1000) = 400\mathcal{C}(10) + 19(59\mathcal{C}(10)) = (400 + 1121)\mathcal{C}(10) = 1521\mathcal{C}(10)$$

★★★

Exercise 3
Design a Clos network, rearrangeable, 100×100*, using modules* 10×10*. Compute the final complexity in function of* $\mathcal{C}(10)$*.*

Solution:

$$N = 100$$
$$p = 10$$
$$q = \frac{N}{p} = 10$$
$$l = p = 10$$

The resulting network is shown in figure 1.4.

$$\mathcal{C}_{REARR}(100) = 30\mathcal{C}(10)$$

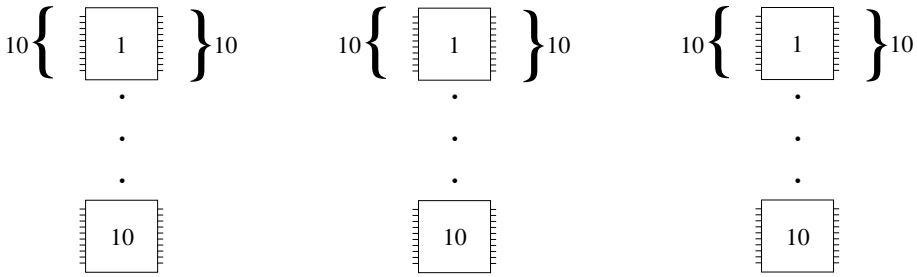


Figure 1.4: Rearrangeable Clos network 100×100

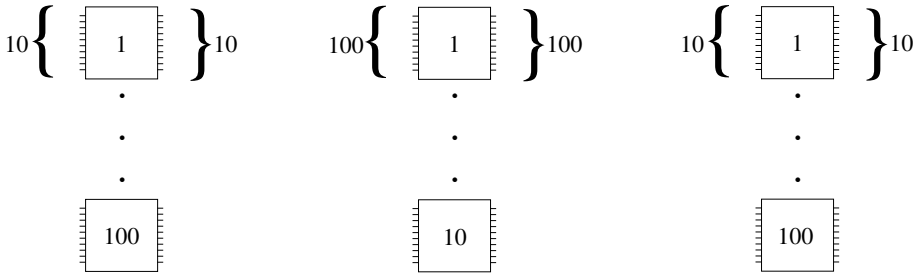


Figure 1.5: Rearrangeable Clos network 1000×1000

★★★

Exercise 4
Design a Clos network, rearrangeable, 1000×1000*, using modules* 10×10*. Compute the final complexity in function of* $\mathcal{C}(10)$*.*

Solution: Rearrangeable (REARR) Clos network, total inputs and outputs $N = 1000$, 10×10 modules.

$$p = 10$$
$$q = \frac{N}{p} = 100$$
$$l = p = 10$$

The resulting network is shown in figure 1.5.

$$\mathcal{C}_{REARR}(1000) = 200\mathcal{C}(10) + 10\mathcal{C}_{REARR}(100) = (200 + (10 \times 30))\mathcal{C}(10) = 500\mathcal{C}(10)$$

★★★

Exercise 5
Consider a Clos network, rearrangeable 9×9*, with* $p = 3$*, and the following Paull matrix:*

$$\begin{bmatrix} - & b & - \\ - & a, c & b \\ c & - & a \end{bmatrix}$$

being the modules of the second stage a*,* b *e* c*.*

1. *Draw the active connections in the network and write a possible set of input/output connections, satisfying the Paull matrix.*

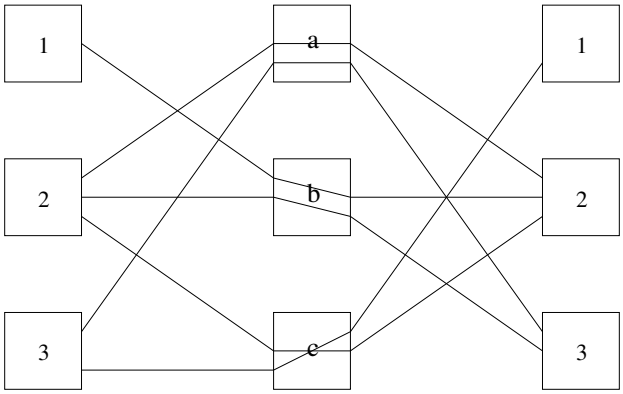


Figure 1.6: Active connections according to the initial Paull matrix

$INPUT$	$OUTPUT$
1	4
4	7
5	5
6	6
8	3
9	8

2. *Connect module 1 of the first stage with module 1 of the third stage. Recompute the Paull matrix and draw the corresponding connections. Should the network be reconfigured? Is the solution unique?*

3. *Connect again module 1 of the first stage with module 1 of the third stage. Recompute the Paull matrix and draw the corresponding connections. Should the network be reconfigured? Is the solution unique?*

Solution:

1. Figure 1.6 shows the network with the active connections of the initial Paull matrix. A possible set of input/output connections is the following:

2. No, there exists an unique solution and the network is not reconfigured. The Paull matrix becomes:

$$\begin{bmatrix} a & b & - \\ - & a,c & b \\ c & - & a \end{bmatrix}$$

   The final network is shown in figure 1.7.

3. Yes, in this case the network is reconfigured and there exist two possible solutions. The first corresponds to P_1 Paull matrix:

$$P_1 = \begin{bmatrix} a,c & b & - \\ - & a,c & b \\ b & - & a \end{bmatrix}$$

   The final network is shown in figure 1.8. The second solution corresponds to P_2 Paull

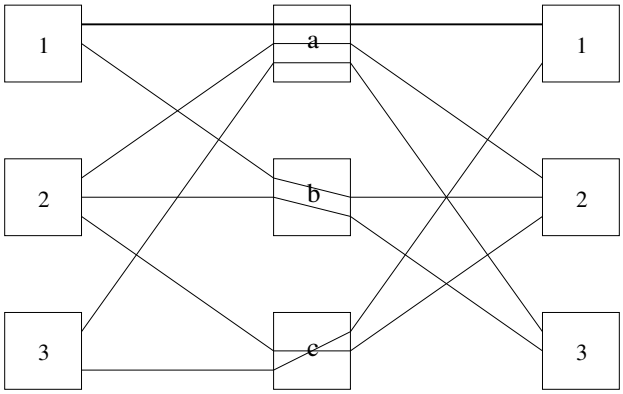


Figure 1.7: Active connections according to the new Paull matrix

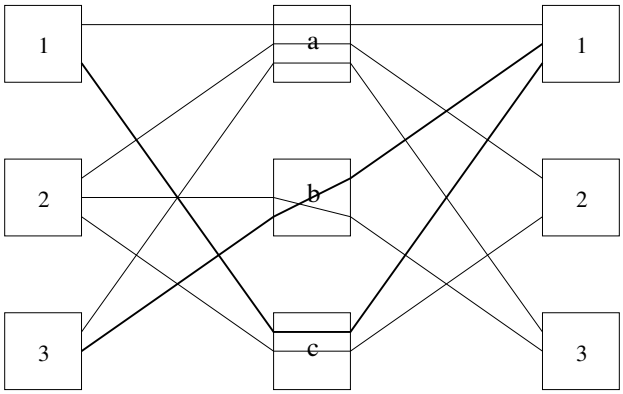


Figure 1.8: Network corresponding to P_1

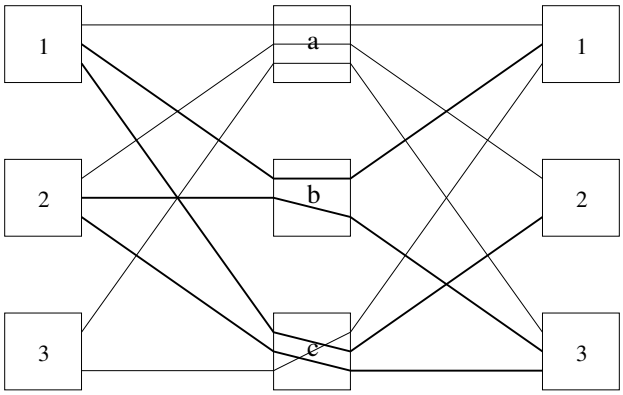


Figure 1.9: Network corresponding to P_2

matrix:

$$P_2 = \begin{bmatrix} a,b & c & - \\ - & a,b & c \\ c & - & a \end{bmatrix}$$

The final network is shown in figure 1.9.

★★★

Exercise 6
Consider a rearrangeable 9×9 Clos network, with three modules at the first and third stages, and the following Paull matrix:

$$\begin{bmatrix} - & - & a \\ - & b & c \\ b & a, c & - \end{bmatrix}$$

being the modules of the second stage a, b and c.

1. *Draw the whole network with all the interconnection links.*
2. *Draw the active connections in the network and write a possible set of input/output connections, satisfying the Paull matrix.*
3. *Connect module 1 of the first stage with module 1 of the third stage. Recompute the Paull matrix and draw the corresponding connections. Should the network be reconfigured? Is the solution unique?*
4. *Connect again module 1 of the first stage with module 1 of the third stage. Recompute the Paull matrix and draw the corresponding connections. Should the network be reconfigured? Is the solution unique?*
5. *In such Clos network, independently from the specific Paull matrix above:*
   - *what is the minimum and the maximum number of second-stage modules that can be involved in any rearrangement?*
   - *what is the minimum and the maximum number of pre-existing connections that could be rearranged?*

Solution:
This exercise is almost identical to Ex. 5. Regarding the last two questions, the number of second-stage modules that are involved in any rearrangment is always 2. Thus, the number of pre-existing connections that could be rearranged is between 1 and 4.

★★★

Exercise 7
Design a Clos network, rearrangeable, 24×25 with $n = 6$ and $m = 5$, where n is the number of inputs of the first stage modules and m is the number of outputs of the third stage modules. Consider the following Paull matrix:

$$\begin{bmatrix} - & - & a & b, e & c \\ a, b & d & - & c & - \\ - & c & e, f & - & d \\ d & - & c & a & b, f \end{bmatrix}$$

being a, b, c, d, e and f the modules of the second stage.

1. *Draw the active connections in the network.*
2. *Connect module 1 of the first stage with module 1 of the third stage. Recompute the Paull matrix and draw the corresponding connections. Should the network be reconfigured? Is the solution unique?*
3. *Connect again module 1 of the first stage with module 1 of the third stage. Recompute the Paull matrix and draw the corresponding connections. Should the network be reconfigured? Is the solution unique?*

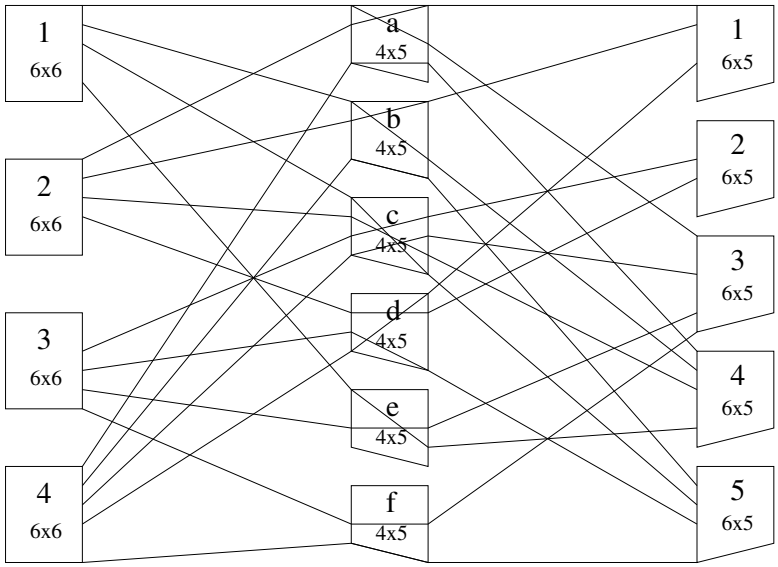


Figure 1.10: Rearrangeable Clos network 24×25

Solution: The number of necessary modules is $r_1 = 4$ and $r_3 = 5$, respectively for the first and third stage.

1. Figure 1.10 shows the network with the active connections corresponding to the initial Paull matrix.

2. No, there exists just one solution for which the network should not be reconfigured. Paul matrix becomes:
$$\begin{bmatrix} f & - & a & b,e & c \\ a,b & d & - & c & - \\ - & c & e,f & - & d \\ d & - & c & a & b,f \end{bmatrix}$$
where it was sufficient to add a link through f to connect the first module of the first stage to the third stage.

3. Yes, in this case the network should be reconfigured. There exist two equivalent solutions: indeed from figure 1.10 it is possible to observe that the required connection can be realized through two different modules (c and d) of the central stage. By rearranging the network and choosing to route the connection through c, the following Paull matrix is obtained:
$$P_1 = \begin{bmatrix} c,f & - & a & b,e & d \\ a,b & c & - & d & - \\ - & d & e,f & - & c \\ d & - & c & a & b,f \end{bmatrix}$$
By choosing instead to route the connection through d, the following Paull matrix is obtained:
$$P_2 = \begin{bmatrix} d,f & - & a & b,e & c \\ a,b & d & - & c & - \\ - & c & e,f & - & d \\ c & - & d & a & b,f \end{bmatrix}$$

★★★

Exercise 8
Compare the complexity of two symmetric Clos networks, the first one that is strictly non blocking and the second one that is rearrangeable. Let N be the number of total ports and p be the number of inputs for the first stage.

1. *Compute the complexity in terms of contact points.*
2. *In the case of the rearrangeable network, compute the value of p minimizing the complexity; what is the final complexity?*
3. *Draw both Clos networks in the cases: $p = 1$ and $p = N$.*

Solution: By setting $q = N/p$ in the formulas of the Clos networks complexity:

$$\mathcal{C}_{\text{SNB}} = (2p-1)(2N + N^2/p^2) \qquad \mathcal{C}_{\text{REARR}} = 2pN + N^2/p$$

Consider now the rearrangeable Clos network. The minimum of $\mathcal{C}_{\text{REARR}}$ is obtained for $\hat{p}$ that can be computed by setting:

$$\frac{\partial \mathcal{C}_{\text{REARR}}}{\partial p} = 2N - \frac{N^2}{p^2} = 0 \qquad \Rightarrow \qquad \hat{p} = \sqrt{\frac{N}{2}}$$

Hence, the minimum complexity is:

$$\mathcal{C}^{opt}_{\text{REARR}} = 2\sqrt{2}N\sqrt{N}$$

In the case $p = 1$, the Clos network degenerates into a crossbar $N \times N$; for $p = N$, the Clos network degenerates into two tandem crossbars. Hence, the complexity for $p = 1$ is equal to N^2 whereas for $p = N$ it is equal to $2N^2$. Note that the optimal complexity is lower in both cases.

★★★

Exercise 9
Design a rearrangeable switch of size 900×450 using only modules of size 10×10, with the aim of minimizing the number of modules.

1. *Describe the architecture*
2. *Compute the total number of modules required*
3. *Describe the configuration algorithm*
4. *Write the formula to compute the minimum theoretical number of modules to build the switch and to compare the actual complexity to the optimal one (question out of scope for the program 2021/22)*

Solution: The 900×450 switch can be built using a Clos network in the following way:

$$C_{900\times450} = 90C_{10} + 10C_{90\times45} + 45C_{10}$$

where the 90×45 switch can be also built using a Clos network:

$$C_{90\times45} = 9C_{10} + 10C_{9\times5} + 5C_{10}$$

in which the last module of the last stage has 5 unconnected outputs. Now observe that a 9×5 switch can be built with a 10×10 module; hence

$$C_{90\times45} = 24C_{10}$$

and finally

$$C_{900\times450} = 375C_{10}$$

Paul's algorithm is used to configure the network, and should be applied recursively twice. The total number of configurations X is

$$X = \frac{900!}{450!}$$

thus the minimum number of modules is

$$\hat{C} = \log_{10!} X = \frac{\log_2(900!) - \log_2(450!)}{\log_2(10!)}$$

Now it is possible to exploit Stirling approximation:

$$\log_2 n! \sim x \log_2 n - 1.44n + 0.5 \log_2 n + 1.32$$

and numerically obtain

$$\hat{C} = 194$$

which is almost half than the actual number of modules adopted for the design.

★★★

Exercise 10
Design a rearrangeable switch of size 600×800 *using only modules of size* 10×10*, with the aim of minimizing the number of modules.*

1. *Describe the architecture.*
2. *Compute the total number of required modules.*
3. *Describe the configuration algorithm.*
4. *Write the formula to compute the minimum theoretical number of modules to build the switch and to compare the actual complexity to the optimal one. (question out of scope for the program 2021/22)*

★★★

Exercise 11
Design a rearrangeable switch of size 400×800 *using only modules of size* 10×10*, with the aim of minimizing the number of modules.*

1. *Describe the architecture.*
2. *Compute the total number of required modules.*
3. *Describe the configuration algorithm.*
4. *Write the formula to compute the minimum theoretical number of modules to build the switch and to compare it with the actual number of adopted modules. (question out of scope for the program 2021/22)*

★★★

Exercise 12
Design a rearrangeable switch of size 500×1000 *using only modules of size* 10×10*, with the aim of minimizing the number of modules.*

1. *Describe the architecture.*
2. *Compute the total number of required modules.*
3. *Describe the configuration algorithm.*

★★★

Exercise 13
Design a rearrangeable switch of size 400×500 *using only modules of size* 10×10*, with the aim of minimizing the number of modules.*

1. *Draw the architecture.*
2. *Compute the total number of required modules.*
3. *What is the simplest routing algorithm that can be adopted?*

Solution:

$$C_{400\times500} = 40 \times C_{10} + 10 \times C_{40\times50} + 50 \times C_{10}$$

Now the middle stage modules can be built as follows:

$$C_{40\times50} = 4 \times C_{10} + 10 \times C_{4\times5} + 5 \times C_{10} = 4 \times C_{10} + 5 \times C_{10} + 5 \times C_{10} = 14 \times C_{10}$$

Hence,

$$C_{400\times500} = 40 \times C_{10} + 10 \times 14 \times C_{10} + 50 \times C_{10} = 230 \times C_{10}$$

★★★

Exercise 14
Design a rearrangeable switch of size 2000×4000 *using only basic modules of size* 10×10*, with the aim of minimizing the number of modules.*

1. *Describe the architecture.*
2. *Compute the total number of required basic modules.*
3. *Describe the configuration algorithm.*
4. *What would have been the approximated number of modules if the switch was designed to be strictly-non-blocking?*

Solution:
In short:

$$C_{4000\times2000} = 600C_{10} + 10C_{400\times200} = 2300C_{10}$$

$$C_{400\times200} = 60C_{10} + 10C_{40\times20} = 170C_{10}$$

$$C_{40\times20} = 11C_{10}$$

The corresponding strictly non-blocking network is expected to have roughly twice the number of basic modules.

★★★

Exercise 15
Design a rearrangeable switch of size 4096×4096 *using only modules of size* 8×8*, with the aim of minimizing the number of modules.*

1. *What is the theoretical minimum number of modules needed? For the sake of easy computation, use the following approximation:* $N! \approx \sqrt{N}N^N$*. (question out of scope for the program 2021/22)*
2. *Describe the proposed architecture.*
3. *Compute the total number of required modules.*
4. *Compare the total number of required modules with the minimum one obtained in question 1. (question out of scope for the program 2021/22)*
5. *Describe how to configure the overall network.*

Solution: One possible solution is to use 8 as factor for the recursive construction. In this case:

$$C_{4096} = 1024C_8 + 8C_{512} = 3584C_8$$
$$C_{512} = 128C_8 + 8C_{64} = 320C_8$$
$$C_{64} = 24C_8$$

Another possible solution is to use $\sqrt{N}$ for the recursive construction. In this case:

$$C_{4096} = 3 \times 64C_{64} = 4608C_8$$
$$C_{64} = 24C_8$$

which is worse than the alternative solution with 8 as factor.

The minimum number of modules $\hat{C}$ can be computed:

$$\hat{C} = \frac{\log_2(4096!)}{\log_2(8!)}C_8$$

Given the approximation:

$$\log_2(N!) = N\log_2 N + 0.5\log_2 N$$

By simple computations (by hand):

$$\log_2(4096!) = 49158 \qquad \log_2(8!) = 25.5$$

and finally:

$$\hat{C} \approx \frac{50000}{25} = 2000C_8$$

Hence, the devised architecture, with factor 8, shows less than twice the number of modules than the minimum one.

★★★

Exercise 16 (out of scope for the program 2021/22)
Consider the design of an asymmetric $N \times \alpha N$ *switch, with* $\alpha \in (0, 1)$*.*

1. *Compute the number of switching configurations supported by the switch.*
2. *What is the complexity reduction with respect to an* $N \times N$ *switch when adopting an optimal theoretical architecture? What is the complexity reduction obtained by adopting the crossbar architecture?*

If needed, use the Stirling approximation:

$$N! \sim \sqrt{2\pi N}\left(\frac{N}{e}\right)^N$$

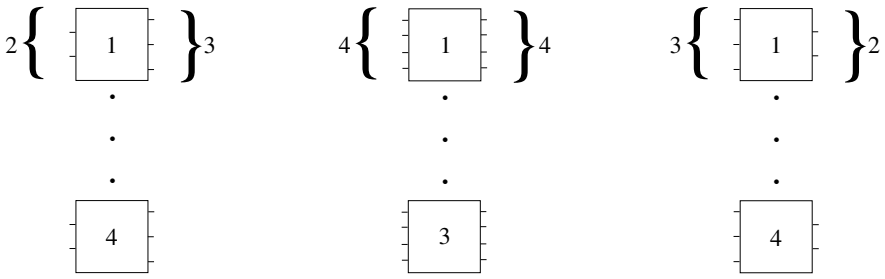


Figure 1.11: Strictly non blocking Clos network: first factorization step

1.1.1 Recursive construction

Exercise 17
Design a $20,000 \times 20,000$ *rearrangeable Clos network, using a recursive construction with a basic building block being a crossbar of size* 10×10*.*

1. *Compute the total number basic building blocks, showing all the involved steps.*

Solution:

$$C_{20000} = 4000C_{10} + 10C_{2000} = (4000 + 12000)C_{10} = 16000C_{10}$$
$$C_{2000} = 400C_{10} + 10C_{200} = (400 + 800)C_{10} = 1200C_{10}$$
$$C_{200} = 40C_{10} + 10C_{20} = (40 + 40)C_{10}$$
$$C_{20} = 4C_{10}$$

★★★

Exercise 18
Design a Clos network, strictly non blocking, of size 8×8*, with two ports for each module of the first stage. Use only modules* 2×2 *through recursive factorization.*

1. *Draw the network at each level of factorization, with all modules and links.*
2. *Draw the final network, with all modules and links.*
3. *Compute formally the complexity of the final network, in function of the complexity* $\mathcal{C}(2)$ *of module* 2×2*.*

Solution: $N = 8$, $p = 2$. At the first factorization step, it will be:

$$p = 2$$
$$q = \frac{N}{p} = 4$$
$$l = 2p - 1 = 3$$

Figure 1.11 shows the network at the first factorization step.

$$\mathcal{C}(8) = 8 \times 2\mathcal{C}(2) + 3\mathcal{C}(4)$$

Note: The complexity of a 2×3 module $= 2\mathcal{C}(2)$. At the second factorization step:

$$p = 2$$

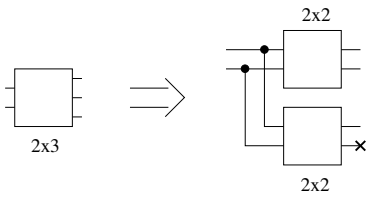


Figure 1.12: Implementation of a 2×3 module

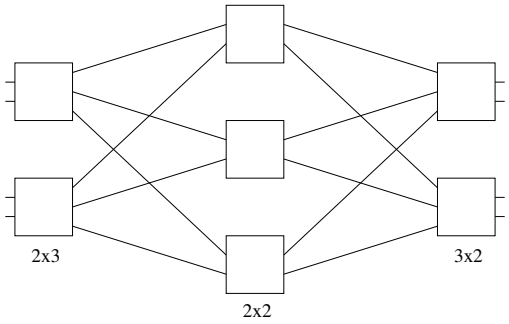


Figure 1.13: Strictly non-blocking 4×4 Clos network

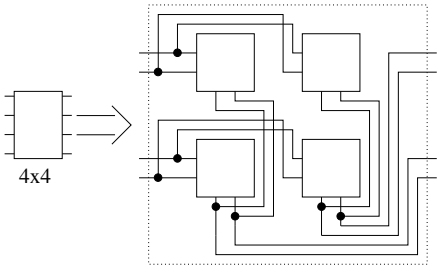


Figure 1.14: Strictly non-blocking 4×4 network

$$q = \frac{N}{p} = 4$$
$$l = 2p - 1 = 3$$

Figure 1.13 shows the network obtained at the second factorization step.

$$\mathcal{C}(4) = 4 \times 2\mathcal{C}(2) + 3\mathcal{C}(2) = 11\mathcal{C}(2)$$

The complexity of the final network is:

$$\mathcal{C}(8) = (16 + (3 \times 11))\mathcal{C}(2) = 49\mathcal{C}(2)$$

As alternative solution, a strictly non-blocking 4×4 network can be built using 4 modules 2×2, according to the scheme presented in Fig. 1.14. Hence,

$$\mathcal{C}(4) = 4\mathcal{C}(2)$$

and in this case the final complexity is:

$$\mathcal{C}(8) = (8 \times 2 + 3 \times 4)\mathcal{C}(2) = 28\mathcal{C}(2)$$

★★★

Exercise 19
Design a Clos network, strictly non blocking, of size 27×27, with three inputs for each module of the first stage. Use only modules 3×3 through recursive factorization.

1. *Draw the network at each factorization level, with all modules and links.*
2. *Draw the final network, with all modules and links.*
3. *Compute formally the complexity of the final network, in function of the complexity $C(3)$ of module 3×3. Show all the involved mathematical steps.*
4. *Which routing algorithm can be used to find a path between an input and an output port?*

Solution: The exercise is similar to problem 18. Here we compute only the complexity. $C(27) = 18C(3 \times 5) + 5C(9)$. Now: $C(3 \times 5) = 2C(3)$; $C(9) = 6C(3 \times 5) + 5C(3) = 12C(3) + 5C(3) = 17C(3)$. Hence, in total $C(27) = 36C(3) + 85C(3) = 121C(3)$.

★★★

Exercise 20
Design a Clos network, rearrangeable, of size $3^h \times 3^h$ with $h = 1, 2, 3, \ldots$, recursively factorized with factor 3, built on 3×3 modules.

1. *Compute formally the network complexity in terms of the number of contact points, in function of h.*
2. *When $h = 3$,*
   (a) *draw the total network, complete of all the interconnections.*
   (b) *show all the steps of the reconfiguration algorithm, connecting: $1 \to 27$, $2 \to 26$, $3 \to 25$, $4 \to 24$, $5 \to 23$, $6 \to 22$.*

★★★

Exercise 21
Consider an $N \times N$ rearrangeable Clos network, with $N = 3^h$ for some integer $h > 1$, factorized recursively with factor 3, and using only 3×3 modules. Let C_3 be a 3×3 module.

1. *Evaluate formally the complexity in terms of C_3 for any h.*

Now consider the specific case $N = 9$.

1. *Draw the whole network.*
2. *Compute the complexity in terms of C_3.*
3. *Connect the following input-output couples: $1 \to 4$, $2 \to 3$, $3 \to 1$, $4 \to 8$, $5 \to 7$, $6 \to 2$, $7 \to 6$, $8 \to 9$, $9 \to 5$.*
4. *Show just the final Paull matrix at the end of the above connections.*

★★★

Exercise 22
Design a Clos network, rearrangeable, symmetric, of size 8×8, in which there exist two modules at the first stage. Through recursive factorization, the network is built around 2×2 modules only.

1. *Draw the final network with all the modules and interconnections.*
2. *Compute the complexity in terms of contact points.*
3. *Draw a Benes network* 8×8*.*
4. *What is the less complex network among the two? Do they have the same blocking probability? Why?*
5. *In the initial Clos network, connect the following input-output couples:* $1 \to 3$, $2 \to 7$, $3 \to 6$, $4 \to 1$, $5 \to 5$, $6 \to 8$, $7 \to 2$, $8 \to 4$*; explain briefly the algorithm used.*

★★★

Exercise 23
Design a rearrangeable Clos network, $N \times N$*, recursively factorized with factor* $\sqrt{N}$*. Assume that the smallest switching module available is* 2×2*.*

1. *Compute formally the complexity in terms of contact points.*
2. *How does the algorithm to configure the network work?*
3. *In the case* $N = 16$*, draw the complete network and highlight all the modules and interconnections.*
4. *Always in the case* $N = 16$*, show the final configuration to connect the following input-output couples, following the algorithm described at question 2: (1,2), (2,10), (3,3), (4,15), (5,16), (6,11), (7,9), (8,1), (9,4), (10,5).*

★★★

Exercise 24
Design a rearrangeable Clos network, of size 1024×1024*, using* 16×16 *modules only.*

1. *Draw the network.*
2. *Compute the final complexity, in function of the complexity of the* 16×16 *modules.*

★★★

Exercise 25
Design an 80×80 *rearrangeable, non-blocking switching fabric, using two possible architectures, the first built with* recursive *factorization of a Clos network with factor* $\sqrt{N}$ *and the second with factor 2.*

1. *Design the two final architectures.*
2. *Compute the complexity in terms of crosspoints for both architectures.*
3. *Discuss the advantages and disadvantages of each of them.*

★★★

Exercise 26
Consider an $N \times N$ *rearrangeable Clos network, factorized recursively with factor 3, and using only* 3×3 *modules. Let* C_n *be the number of* $n \times n$ *modules. We assume* N *to be a power of 3.*

1. *evaluate formally the complexity in terms of* C_3 *and in terms of number of crosspoints*

2. *for $N \to \infty$, compare the complexity with the Benes network, in both cases*[1] *:*
   - *in terms of number of elementary modules (i.e., $C_3 = C_2$)*
   - *in terms of number of crosspoints*

Now consider the case $N = 9$.

1. *draw the whole network*
2. *connect the following input-output couples, showing the final Paull matrix:* $1 \to 4$, $2 \to 9$, $3 \to 1$, $4 \to 7$, $5 \to 3$, $6 \to 2$, $7 \to 8$, $8 \to 6$, $9 \to 5$,

Solution: From the recursive factorization we have that $p = 3$ and $q = N/p = N/3$. The resulting rearrangeable Clos network is shown in Figure 1.15. From the figure it is easy to see that the complexity is:

$$\begin{aligned} C_{3\times3}(N) &= \frac{N}{3} + 3C\left(\frac{N}{3}\right) + \frac{N}{3} = \frac{2}{3}NC_3 + 3C\left(\frac{N}{3}\right) \\ &= \frac{2}{3}NC_3 + 3\left[\frac{2}{3}\frac{N}{3}C_3 + 3C\left(\frac{N}{3^2}\right)\right] \\ &= \frac{2}{3}NC_3 + \frac{2}{3}NC_3 + 3^2C\left(\frac{N}{3^2}\right) = \frac{2}{3}NC_3 + \frac{2}{3}NC_3 + 3^2\left[\frac{2}{3}\frac{N}{3^2}C_3 + 3C\left(\frac{N}{3^3}\right)\right] \\ &= \frac{2}{3}NC_3 + \frac{2}{3}NC_3 + \frac{2}{3}NC_3 + 3^3C\left(\frac{N}{3^3}\right) = \frac{2}{3}NkC_3 + 3^kC\left(\frac{N}{3^k}\right) \end{aligned} \quad (1.1)$$

The recursion stops when $N/3^k = 3$. Therefore $3^k = N/3$ and $k = \log_3 N - 1$. Substituting in (1.1):

$$\begin{aligned} C_{3\times3}(N) &= \frac{2}{3}N(\log_3 N - 1)C_3 + \frac{N}{3}C_3 = \frac{2}{3}N\log_3 NC_3 - \frac{N}{3}C_3 \\ &= \left(\frac{2}{3}N\log_3 N - \frac{N}{3}\right)C_3 \end{aligned} \quad (1.2)$$

The complexity in terms of 3×3 modules is, from (1.2):

$$C^m_{3\times3}(N) = \frac{2}{3}N\log_3 N - \frac{N}{3} = \frac{2}{3}N\frac{\log_2 N}{\log_2 3} - \frac{N}{3} \approx 0.42N\log_2 N - \frac{N}{3}$$

which, for $N \to \infty$, tends to

$$C^m_{3\times3}(N) \to 0.42N\log_2 N \quad (1.3)$$

In terms of crosspoints ($C_3 = 9$):

$$C^{cp}_{3\times3}(N) = 9\left(\frac{2}{3}N\log_3 N - \frac{N}{3}\right) = 6N\frac{\log_2 N}{\log_2 3} - 3N \approx 3.80N\log_2 N - 3N$$

which, for $N \to \infty$, tends to

$$C^{cp}_{3\times3}(N) \to 3.80N\log_2 N \quad (1.4)$$

Recall that Benes network complexity is $C(N) = (N\log_2 N - \frac{N}{2})C_2$. Therefore ($C_2 = 4$):

$$C^m_{\text{Benes}}(N) = N\log_2 N - \frac{N}{2} \to N\log_2 N \quad (1.5)$$

$$C^{cp}_{\text{Benes}}(N) = 4N\log_2 N - 2N \to 4N\log_2 N \quad (1.6)$$

[1] Recall that $\log_2 3 \approx 1.58$

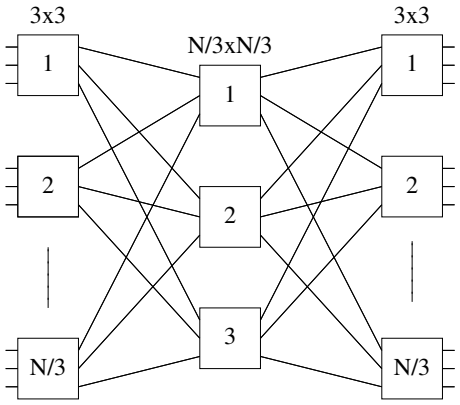


Figure 1.15: (Ex. 26) Rearrangeable Clos network, $N \times N$, factorized recursively

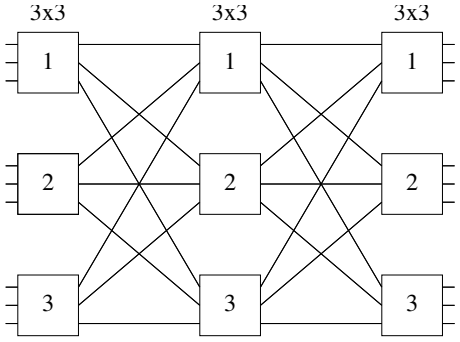


Figure 1.16: (Ex. 26) Rearrangeable Clos network, 9×9

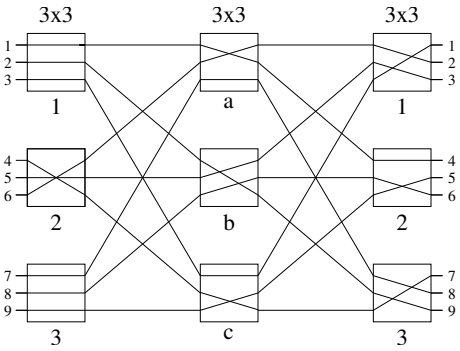


Figure 1.17: (Ex. 26) Clos network configured by Paul algorithm

Combining (1.3) with (1.5) and (1.4) with (1.6):

$$\frac{C^m_{3\times3}}{C^m_{\text{Benes}}} \approx 0.42$$
$$\frac{C^{cp}_{3\times3}}{C^{cp}_{\text{Benes}}} \approx 0.95$$

Hence, for $N \to \infty$ the rearrangeable Clos network uses less then half the modules as the Benes network, but each 3×3 module is more complex than a 2×2 module so the number of crosspoints is almost equal.

The whole network with $N = 9$ is shown in Fig. 1.16. The connections are configured using the Paull algorithm. The final Paull matrix is the following:

$$\begin{bmatrix} c & a & b \\ a,b & - & c \\ - & b,c & a \end{bmatrix}$$

which corresponds to the network configuration of Fig. 1.17.

★★★

Exercise 27
Consider a $N \times N$ rearrangeable non-blocking network, with $N = 4^h$ and $h = 1, 2, \ldots,$. The network is built according to one of two design architectures:

- *recursive factorization with basic modules of size 2×2;*
- *recursive factorization with basic modules of size 4×4.*

Answer the following:

1. *Compute formally the complexity in terms (i) of basic modules and (ii) of crosspoints, for both architectures: write the proper recursive equation and solve it.*
2. *Which architecture is optimal from the complexity point of view?*
3. *Which set of algorithms can be used to configure the connections in each architecture?*

Solution: In the case of basic modules of size 2×2, the network is a Benes network, for which we know that the number of modules is:

$$C(N) = \left(N \log_2 N - \frac{N}{2} \right) C_2$$

and the corresponding number of crosspoints is:

$$X(N) = 4N \log_2 N - 2N$$

In the case of basic modules of size 4×4, the number of modules satisfies:

$$C(N) = 2\frac{N}{4} C_4 + 4C\left(\frac{N}{4}\right) \tag{1.7}$$

and in general

$$C\left(\frac{N}{4^k}\right) = 2\frac{N}{4^{k+1}} C_4 + 4C\left(\frac{N}{4^{k+1}}\right) \tag{1.8}$$

By using (1.8) to unfold (1.7):

$$C(N) = \frac{k}{2} N C_4 + 4^k C\left(\frac{N}{4^k}\right) \qquad \text{for } k = 1, \ldots, \frac{1}{2}\log_2 N - 1 \tag{1.9}$$

Now (1.9) becomes:

$$C(N) = \left(\frac{1}{4}\log_2 N - \frac{1}{2}\right) N C_4 + \frac{N}{4} C_4 = \left(\frac{N}{4}\log_2 N - \frac{N}{4}\right) C_4 \tag{1.10}$$

and the corresponding number of crosspoints is:

$$X(N) = 4N \log_2 N - 4N$$

By comparing both the complexity in terms of basic modules and number of crosspoints, the architecture with basic modules of size 4×4 is the most convenient.

The algorithms to configure the Benes networks are the looping algorithm and the Paull's algorithm, whereas to configure the other network it is necessary to use the Paull's algorithm.

★★★

Exercise 28

Consider a $N \times N$ rearrangeable Clos network with basic modules of size 10×10. Let $N = 10^{2^k}$ for some integer $k = 1, 2, \ldots$ The cost is evaluated in terms of number of basic modules. Consider now two alternative architectures: (A1) in which the network is built recursively with factor 10; (A2) in which the network is built recursively with factor $\sqrt{N}$.

1. *Evaluate formally the cost C_1, in function of N, for (A1)*
2. *Evaluate formally the cost C_2, in function of N, for (A2)*
3. *Compare C_1 and C_2*
4. *Which control algorithm can be used for (A1)?*
5. *Which control algorithm can be used for (A2)?*
6. *Can (A1) and/or (A2) be used in the Automatic Main Distribution Frame in the central office connecting the ADSL lines to the DSLAM? Why? (question out of scope for the program 2021/22)*

Solution:

Let $n = 2^k$ and $N = 10^n$.

$$C_1(10^n) = 2 \times 10^{n-1} C_{10} + 10 C(10^{n-1}) = 2h 10^{n-1} C_{10} + 10^h C(10^{n-h})$$

for $h = 0, \ldots, n-1$. By setting $h = n - 1$:

$$C_1(10^n) = 2(n-1)10^{n-1} C_{10} + 10^{n-1} C_{10} = (2n-1)10^{n-1} C_{10}$$

from which:

$$C(N) = (2\log_{10} N - 1)\frac{N}{10} C_{10}$$

To evaluate C_2, it is equivalent to slide 28 of Part II. The only difference if that

$$C(10^n) = 3 \times 10^{n/2} C(10^{n/2}) \approx 3^h 10^n C(10^{n/2^h})$$

$$C_2(N) \approx 3^{\log_2 n} 10^n C_{10} = n^{\log_2 3} N C_{10} = (\log_{10} N)^{\log_2 3} N C_{10}$$

Both are configured using Paull algorithm.

Both cannot be applied in a AMDF because of possible reconfigurations, unless accepting some interruptions.

★★★

Exercise 29
Design a rearrangeable switch of size $10,000 \times 10,000$ *using only modules of size* 10×10*, with the aim of minimizing the number of modules.*

1. *Describe the architecture.*
2. *Compute the total number of required modules.*
3. *Describe the configuration algorithm.*
4. *Write the formula to compute the minimum theoretical number of* 10×10 *modules. Note that the formula must be compatible with a standard scientific calculator. Recall the Stirling approximation:* $N! \approx \sqrt{N} N^N$*. (question out of scope for the program 2021/22)*

Solution:
Exploiting a standard recursive construction:

$$C_{10000} = 2000C_{10} + 10C_{1000} = (2000 + 10 \times 500)C_{10} = 7000C_{10}$$

being

$$C_{1000} = 200C_{10} + 10C_{100} = (200 + 300)C_{10} = 500C_{10}$$

being

$$C_{100} = 30C_{10}$$

The minimum number of modules can be computed as:

$$\hat{C}_{10} = \log_{10!}(10000!) = \frac{\log(10000!)}{\log(10!)} \approx \frac{0.5\log(10000) + 10000\log(10000)}{0.5\log(10) + 10\log(10)}$$

Using a calculator:

$$\hat{C}_{10} = 3809$$

which implies that the proposed solution is only $(7000 - 3809)/3809 = 83\%$ larger than the lower bound.

1.1.2 Non-interruptible networks

Exercise 30 (out of scope for the program 2021/22)
Consider a 2000×2000 *switch built just using basic modules* 10×10*.*

1. *Design a minimum-cost strictly non-blocking network. Draw the structure of the network and compute the total number of basic modules.*
2. *Design a minimum-cost non-interruptible, rearrangeable switching network. Draw the structure of the network and compute the total number of basic modules.*
3. *Are the two networks equivalent in terms of cost (i.e., number of basic modules) and control (i.e., configuration algorithm)? Why?*

Solution: To design a strictly-non-blocking network,

$$C_{2000,SNB} = 200C_{10\times 19} + 19C_{200,SNB} + 200C_{19\times 10}$$

where

$$C_{10\times 19} = 2C_{10}$$

$$C_{200,SNB} = 20C_{10\times19} + 19C_{20,SNB} + 20C_{10\times19}$$

and

$$C_{20,SNB} = 4C_{10}$$

since 4 crossbar $k \times k$ can be always combined to build a crossbar $(2k) \times (2k)$. Hence,

$$C_{200,SNB} = (20 \times 2 + 19 \times 4 + 20 \times 2)C_{10} = 156C_{10}$$

and, finally,

$$C_{2000,SNB} = (200 \times 2 + 19 \times 156 + 200 \times 2)C_{10} = 3764C_{10}$$

To design a non-interruptible, rearrangeable (NIR) network,

$$C_{2000,NIR} = 250C_{8\times10} + 10C_{250,REAR} + 250C_{10\times8}$$

where

$$C_{8\times10} = C_{10}$$

Hence,

$$C_{250,REAR} = 25C_{10} + 10C_{25,REAR} + 25C_{10}$$

where, using the classical Clos construction:

$$C_{25,REAR} = 3C_9 + 9C_3 + 3C_9$$

where

$$C_9 = C_{10} \quad C_3 = C_{10}/3$$

Finally,

$$C_{25,REAR} = 9C_{10}$$

and

$$C_{250,REAR} = 140C_{10}$$

and

$$C_{2000,NIR} = 1900C_{10}$$

The first network has a complexity almost twice than the second one, but its control algorithm is trivial. The control algorithm for the second network is a variant of Paul algorithm, in which the two additional medium-stage modules are used to exploit multipath and avoid interruptions.

★★★

Exercise 31 (out of scope for the program 2021/22)
Consider a 3000×1000 *switch built just using basic modules* 10×10*.*

1. *Design a minimum-cost, strictly non-blocking network. Draw the structure o f the network and compute the total number of basic modules.*
2. *Design a minimum-cost non-interruptible, rearrangeable switching network. Draw the structure of the network and compute the total number of basic modules.*
3. *Are the two networks equivalent in terms of cost (i.e., number of basic modules) and control (i.e., configuration algorithm)? Why?*

Solution:

1. For the SNB network:

$$C_{SNB}(3000 \times 1000) = 300C(10 \times 19) + 19C_{SNB}(300 \times 100) + 100C(19 \times 10)$$

where

$$C_{SNB}(300 \times 100) = 30C(10 \times 19) + 19C(30 \times 10) + 10C(19 \times 10)$$

where $C(30 \times 10) = 3C(10)$. Hence, $C_{SNB}(300 \times 100) = 137C(10)$ and $C_{SNB}(3000 \times 1000) = 3403C(10)$.

2. For the NIR network, two possible architectures are possible.

   (a) One solution is:

$$C_{NIR}(3000 \times 1000) = 375C(8 \times 10) + 10C_{REAR}(375 \times 125) + 125C(10 \times 8)$$

   where $C(8 \times 10) = C(10)$ and

$$C_{REAR}(375 \times 125) = 38C(10) + 10C(38 \times 13) + 13C(10)$$

   where $C(38 \times 13) = 8C(10)$ thanks to a "grid" architecture. Hence, $C_{REAR}(375 \times 125) = 131C(10)$ and $C_{NIR}(3000 \times 1000) = 1810C(10)$.

   (b) Another solution is:

$$C_{NIR}(3000 \times 1000) = 300C(10 \times 12) + 12C_{REAR}(300 \times 100) + 100C(12 \times 10)$$

   where $C(12 \times 10) = 2C(10)$ and

$$C_{REAR}(300 \times 100) = 30C(10) + 10C(30 \times 10) + 10C(10)$$

   where $C(30 \times 10) = 3C(10)$. Hence, $C_{REAR}(300 \times 100) = 70C(10)$ and $C_{NIR}(3000 \times 1000) = 1640C(10)$.

3. By comparing the final cost of SNB and NIR networks, NIR networks show a smaller complexity (about a factor 2) in terms of basic modules. On the contrary, the control for a SNB network is much simpler than the one for a NIR network, since it is just based on computing a generic path between the required source and destination (e.g, using a breath first search algorithm), whereas NIR networks require to run Paul algorithm and exploit double paths for the first and the last stages of the network.

★★★

Exercise 32 (out of scope for the program 2021/22)
Consider a 1000×1000 *switch built just using basic modules* 10×10*.*

1. *Design a minimum-cost, strictly non-blocking network. Draw the structure of the network and compute the total number of basic modules.*
2. *Design a minimum-cost non-interruptible, rearrangeable switching network. Draw the structure of the network and compute the total number of basic modules.*
3. *Are the two networks equivalent in terms of cost (i.e., number of basic modules) and control (i.e., configuration algorithm)? Why?*

Solution:

For the SNB:

$$C_{1000} = 200C_{10\times 19} + 19C_{100} = 400C_{10} + 19C_{100} = (400 + 19 \times 59)C_{10} = 1521C_{10}$$

being

$$C_{100} = 20C_{10\times 19} + 19C_{10} = 59C_{10}$$

For the NIR, the first possibile solution is:

$$C_{1000,NIR} = 10C_{125,REAR} + 250C_{8\times 10} = (10 \times 66 + 250)C_{10} = 910C_{10}$$

being

$$C_{125,REAR} = 26C_{10} + 10C_{13} = (26 + 10 \times 4)C_{10} = 66C_{10}$$

being

$$C_{13} = 4C_{10}$$

For the NIR, the second possible solution is:

$$C_{1000,NIR} = 200C_{10\times 12} + 12C_{100,REAR} = (400 + 12 \times 30)C_{10} = 760C_{10}$$

being

$$C_{100,REAR} = 30C_{10}$$

1.2 Benes networks

Exercise 33

For a 8×8 Benes network, use the looping algorithm to configure the network and connect the input output couples in table 1.1.

$INPUT$	$OUTPUT$
1	6
2	4
5	8
6	3
7	2
8	5

Table 1.1: Connections to configure in Ex. 33

Solution: As shown in figure 1.18, a 8×8 Benes network has 5 stages, each of them built by 4 2×2 modules. The input sequence chosen to configure is: $1.start_{in} = 2$; $2.start_{in} = 8$; $3.start_{in} = 7$. The result after this first step of the algorithm is shown in figure 1.19.

Then, the looping algorithm is applied in the internal modules of the network. For the upper central module, the chosen input sequence is: $1.start_{in_1} = 2$; $2.start_{in_1} = 8$. The final configuration is shown in figure 1.20. For the lower central module, the chosen input sequence is: $1.start_{in_1} = 1$; $2.start_{in_1} = 6$. The result is shown in figure 1.21. The final result is given by the union of the three steps of the algorithm and is shown in figure 1.22.

★★★

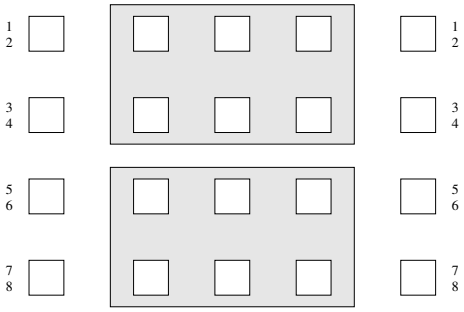


Figure 1.18: Benes network without connections

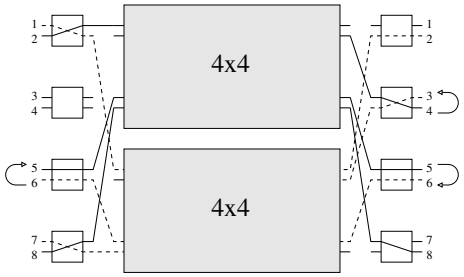


Figure 1.19: After the first step of the looping algorithm. Continuous lines refer to forward connections, whereas dot lines refer to backward connections.

Figure 1.20: After the second step of the looping algorithm, applied to the upper central module

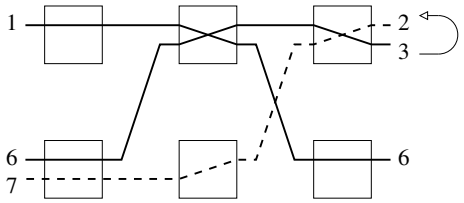


Figure 1.21: Result of the second step of the looping algorithm, applied to the lower central module

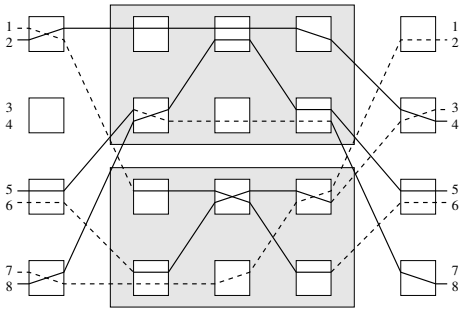


Figure 1.22: Final result of the looping algorithm

$INPUT$	$OUTPUT$
1	3
2	6
3	2
4	1
5	8
6	4
7	5
8	7

Table 1.2: Input-output couples to connect

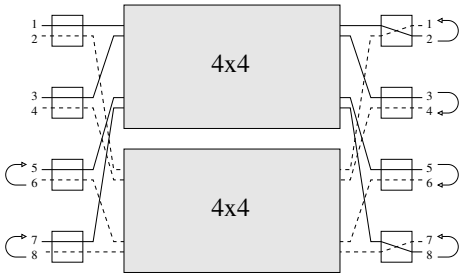


Figure 1.23: Results after the first step of the looping algorithm

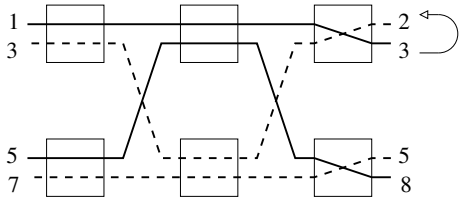


Figure 1.24: Results after the second step of the looping algorithm on the upper central section

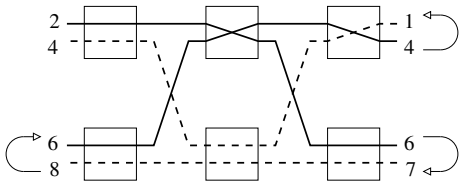


Figure 1.25: Results after the second step of the looping algorithm on the lower central section

Exercise 34
For a 8×8 Benes network, use the looping algorithm to connect the input-output coupled shown in table 1.2.

Solution: The initial network without connections is shown in Fig. 1.18. Now apply the *looping* algorithm; the starting inputs are: $1.start_{in} = 1$; $2.start_{in} = 3$. The result of this first step is shown in Fig. 1.23. Then the looping algorithm is applied on both the central sections of the network. For the upper section, the starting inputs are: $1.start_{in_1} = 1$; $2.start_{in_1} = 5$. The result is shown in Fig. 1.24. For the lower section, the starting inputs are: $1.start_{in_1} = 2$. The result is shown in Fig. 1.25. The final result is given by collecting the three partial results above

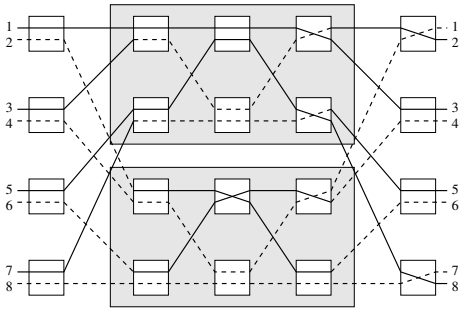


Figure 1.26: Final result of the looping algorithm

and it is shown in Fig. 1.26.

★★★

Exercise 35
Design an 8×8 Benes network. Connect the following input-output couples: $1 \to 3$, $2 \to 7$, $3 \to 1$, $4 \to 2$, $5 \to 4$, $7 \to 6$, $8 \to 5$.

1. *Draw the complete network, showing all the recursively built modules.*
2. *Use the looping algorithm to configure the network. Show graphically the used loops, assuming that the inputs are always considered in increasing sequence.*
3. *Show the final configuration of the network, after running the looping algorithm.*
4. *Is it possible to use Paull algorithm to configure the network? Why? If true, use it showing the evolutions of Paull matrix at each addition of a new connection; the connection sequence to choose is given by the increasing number of inputs.*
5. *Draw the final configuration of the network, after running the Paull algorithm.*
6. *Is there any difference between the final configurations obtained by the two algorithms? Is it always like this? Why?*

Solution: For the first 4 questions, the exercise is the same as exercise 33. The loops to consider are the following (using the notation "in-out"):

- 8×8 switch: (1-3, 5-4, 2-7); (3-1, 4-2); (7-6, 8-5). Starting inputs: 1,3,7.
- Upper 4×4 switch, with connections $1 \to 2$, $2 \to 1$, $4 \to 3$: (1-2, 2-1); (4-3). Starting inputs: 1,4.
- Lower 4×4 switch, with connections $1 \to 4$, $2 \to 1$, $3 \to 2$, $4 \to 3$: (1-4, 4-3, 3-2, 2-1). Starting input: 1.

It is possible to use Paull algorithm, since the Benes network is a particular rearrangeable Clos network. For the 8×8 switch, adding the connections $1 \to 3$, $2 \to 7$, $3 \to 1$, $4 \to 2$, $5 \to 4$,

$7 \to 6$, $8 \to 5$, the sequence of Paull matrices is:

$$1 \to 3 : \begin{bmatrix} - & U & - & - \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix} ; \; 2 \to 7 : \begin{bmatrix} - & U & - & D \\ - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix} ; \; 3 \to 1 : \begin{bmatrix} - & U & - & D \\ U & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix} ;$$

$$4 \to 2 : \begin{bmatrix} - & U & - & D \\ U, D & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix} ; \; 5 \to 4 : \begin{bmatrix} - & U & - & D \\ U, D & - & - & - \\ - & D & - & - \\ - & - & - & - \end{bmatrix} ;$$

$$7 \to 6 : \begin{bmatrix} - & U & - & D \\ U, D & - & - & - \\ - & D & - & - \\ - & - & U & - \end{bmatrix} ; \; 8 \to 5 : \begin{bmatrix} - & U & - & D \\ U, D & - & - & - \\ - & D & - & - \\ - & - & U, D & - \end{bmatrix}$$

For the upper 4×4 switch, adding the connections $1 \to 2$, $2 \to 1$, $4 \to 3$, the sequence of the Paull matrices is the following:

$$1 \to 2 : \begin{bmatrix} U & - \\ - & - \end{bmatrix} ; \; 2 \to 1 : \begin{bmatrix} U, D & - \\ - & - \end{bmatrix} ; \; 4 \to 3 : \begin{bmatrix} U, D & - \\ - & D \end{bmatrix} ;$$

For the lower 4×4 switch, adding the connections $1 \to 4$, $2 \to 1$, $3 \to 2$, $4 \to 3$, the sequence of the Paull matrices is the following:

$$1 \to 4 : \begin{bmatrix} - & U \\ - & - \end{bmatrix} ; \; 2 \to 1 : \begin{bmatrix} D & U \\ - & - \end{bmatrix} ; \; 3 \to 2 : \begin{bmatrix} D & U \\ U & - \end{bmatrix} ; \; 4 \to 3 : \begin{bmatrix} D & U \\ U & D \end{bmatrix}$$

In this case, the final configuration obtained through Paull algorithm is the same as the one through looping algorithm. It is just a chance, since other decision choices (when possible) might result in another configuration.

★★★

Exercise 36

Draw the complete 16×16 *Benes network. Describe the algorithm to configure the network.*

Coherently with this algorithm, connect the following input-output couples: $1 \to 6$, $2 \to 5$, $3 \to 16$, $4 \to 7$, $5 \to 15$, $6 \to 1$, $7 \to 8$, $8 \to 14$, $9 \to 9$, $10 \to 2$, $11 \to 10$, $12 \to 3$, $13 \to 11$, $14 \to 4$, $15 \to 12$, $16 \to 13$.

★★★

Exercise 37

Consider a $N \times (N/2)$ *rearrangeable switch, acting as concentrator, built with* 2×2 *basic modules exploiting recursive factorization. Assume* $N = 2^h$*, for some positive integer* $h \geq 2$*.*

1. *Compute formally the total number of basic modules.*
2. *Compute the minimum theoretical number of modules necessary to implement the switch and compare with the actual complexity of the designed architecture. If needed, use Stirling approximation. (question out of scope for the program 2021/22)*

$$\log_2 n! \sim n \log_2 n - 1.44n + 0.5 \log_2 n + 1.32$$

3. *Compare the actual complexity with the one achieved designing a symmetric* $N \times N$ *Benes network and then removing the useless modules just at the last stage of the whole network*

4. *Consider the control algorithm to configure the $N \times (N/2)$ switch.*

   (a) Can the Paul algorithm be used? Why?

   (b) Can the looping algorithm be used? Why?

Solution:

1. Let $C(N_1, N_2)$ be the complexity of a $N_1 \times N_2$ switch. Let C_2 be a basic 2×2 module. Thus, for any $k = 1, \ldots, \log_2 N - 1$:

$$C(N, N/2) = \frac{N}{2}C_2 + \frac{N}{4}C_2 + 2C(N/2, N/4) = k\frac{3}{4}NC_2 + 2^k C(N/2^k, N/2^{k+1})$$

   Hence,

$$C(N, N/2) = \frac{3}{4}N(\log_2 N - 1)C_2 + \frac{N}{2}C_2 = \left(\frac{3}{4}N\log_2 N - \frac{N}{4}\right)C_2$$

2. The total number of switching configurations is

$$X = \frac{N!}{\left(\frac{N}{2}\right)!}$$

   and thus the minimum number of 2×2 modules is

$$\hat{C}(N, N/2) = \log_2 X = \log_2(N!) - \log_2((N/2)!) \sim$$
$$N\log_2 N - 1.44N - \frac{N}{2}\log_2\frac{N}{2} - 1.44\frac{N}{2} = \frac{N}{2}\log_2 N - 1.66N$$

   Thus the relative complexity of the proposed solution is

$$\frac{C(N, N/2)}{\hat{C}(N, N/2)} \to \frac{3}{2}$$

   i.e. it requires 50% more modules than the minimum theoretical one.

3. The complexity of reduced Benes network $C'(N, N/2)$ is obtained by removing $N/4$ modules in the last $(\log_2 N - 1)$ stages and hence the final complexity is:

$$C'(N, N/2) = \left(N\log_2 N - \frac{N}{2}\right)C_2 - \frac{N}{4}(\log_2 N - 1)C_2 = \left(\frac{3}{4}N\log_2 N - \frac{N}{4}\right)C_2$$

   which is perfectly equivalent to the previous solution.

4. Paul algorithm can be used, in a recursive way, since the switch is rearrangeable. Looping algorithm can be also used, in a recursive way, since the number of modules in the middle stage is always two for each level of recursion.

★★★

Exercise 38

Consider a $N \times (N/2)$ rearrangeable switch, acting as concentrator, built with 2×2 basic modules, exploiting recursive factorization. Assume $N = 2^h$, for some positive integer $h \geq 2$.

1. *Consider a first architecture with recursive factorization.*

(a) *Compute the total number of basic modules, showing all the steps to solve the corresponding recursive equation.*

(b) *Draw the complete network topology for a* 8×4 *switch and compute the total number of basic modules.*

(c) *Can Paul and looping algorithms be used to configure the network? Why?*

2. *Consider a second architecture built by designing a symmetric* $N \times N$ *Benes network and then removing the useless modules starting from at the last stage.*

(a) *Compute the total number of basic modules (starting from the formula for Benes networks).*

(b) *Draw the complete network topology for a* 8×4 *switch and compute the total number of modules.*

(c) *Can Paul and looping algorithms be used to configure the network? Why?*

3. *What is the best among the two architectures?*

Solution:

The problem is almost identical to Ex. 37. In addition:

1.(b) Using recursive constructions, the 8×4 switch has complexity: $C(8,4) = 6C_2 + 2C(4,2) = (6 + 2 \times 5)C_2 = 16C_2$. The corresponding network is shown in Fig. 1.27.

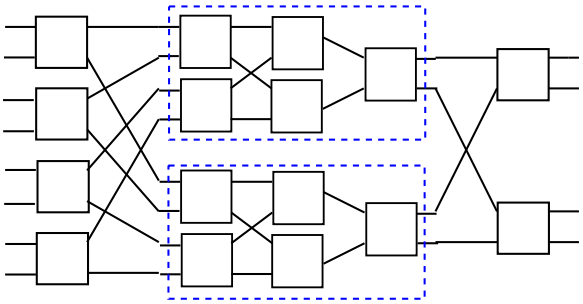


Figure 1.27: Network (b) for Ex. 38

Note that, using a simpler construction, $C(4,2) = 3C_2$ and thus $C(8,4) = 6C_2 + 2C(4,2) = (6 + 2 \times 3)C_2 = 12C_2$. The corresponding network is shown in Fig. 1.28.

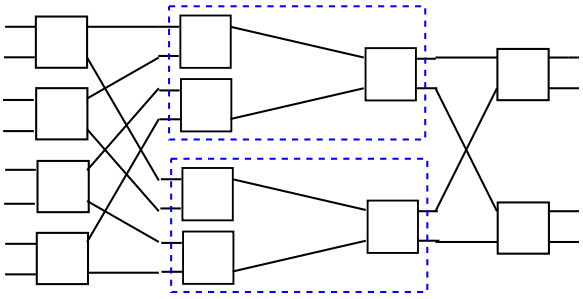


Figure 1.28: Alternative network (b) for Ex. 38

2.(c) Using a 8×8 Benes network and removing 4 useless modules (shown as dotted), we obtain the architecture in Fig. 1.29, whose total complexity is $16C_2$. Thus, the complexity is equivalent to the first architecture.

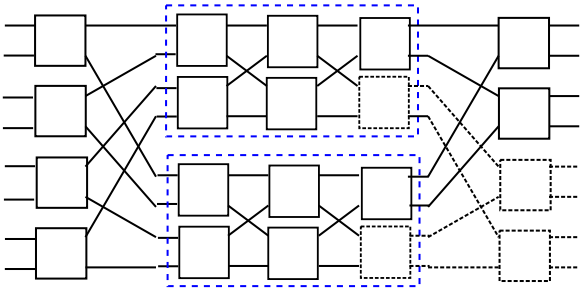


Figure 1.29: Network (c) for Ex. 38

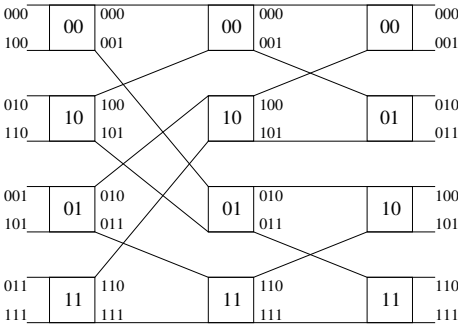


Figure 1.30: Banyan network with Baseline layout (ex. 39)

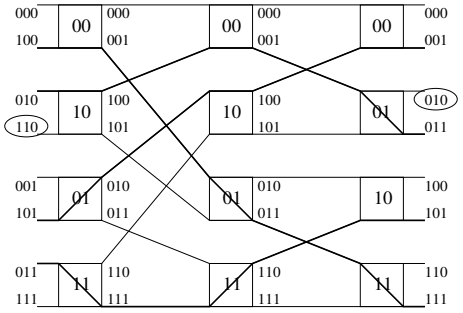


Figure 1.31: Banyan network with Baseline layout - blocking configuration (ex. 39)

1.3 Banyan networks

Exercise 39
Draw an 8×8 Banyan network, having Baseline layout; identify the nodes and the edges of the network. Connect: 2/3, 3/5, 4/7, 5/1, 6/2. Is it possible? Why? Connect: 2/0, 3/2, 4/3, 5/5, 6/7. Is it possible? Why?

Solution: The Banyan network with Baseline layout, with the identifiers, is shown in Fig. 1.30. Fig. 1.31 shows the problem in connecting the first set of links: input 6 cannot be connected with output 2. Note that even the set of inputs is compact, the set of outputs is not monotone, and this configuration does not satisfy the sufficient conditions to avoid blocks. Fig. 1.32 shows the solution for the second set of connections, which now satisfy the sufficient conditions for being non-blocking.

★★★

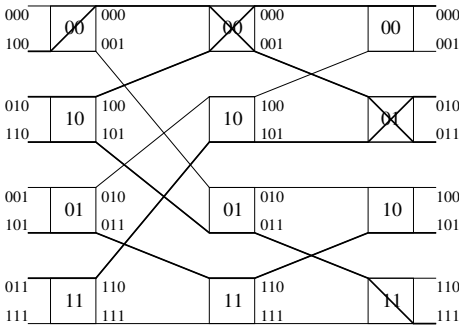


Figure 1.32: Banyan network with Baseline layout - non-blocking configuration (ex. 39)

Exercise 40
Design a 8×8 self-routing multistage network, adopting basic modules of size 2×2.

1. *Draw the whole network with all the interconnections and modules.*
2. *Identify all the modules, input ports, output ports and all the links in order to support self-routing.*
3. *How does the self-routing work?*
4. *What are the properties of the network in terms of blocking?*
5. *Using the Lee method, find the blocking probability in function of the overall traffic λ (measured in Erlang) feeding the whole network. (question out of scope for the program 2021/22)*
6. *Is the blocking probability obtained with Lee compatible with the properties of the networks highlighted in question 4? (question out of scope for the program 2021/22)*

Solution:
The problem is a variant to Ex. 39.
About question 5, the Lee graph is just a sequence of two edges in series. The probability of busy for each edge is $\lambda/8$, thus the blocking probability is $1 - (1 - \lambda/8)^2$.

★★★

Exercise 41
Draw an 8×8 Banyan network, with Shuffle (Ω) layout; identify the nodes and the edges of the network. Connect: 2/0, 3/2, 4/3, 5/5, 6/7.

Solution: See Fig. 1.33 for the solution. Fig. 1.34 shows that the configuration is non-blocking.

★★★

Exercise 42
Draw an 8×8 Banyan network with Banyan layout; identify the nodes and the edges of the network. Connect: 2/0, 3/2, 4/3, 5/5, 6/7.

Solution: See Fig. 1.35 for the solution. Fig. 1.36 shows that the configuration is non-blocking

★★★

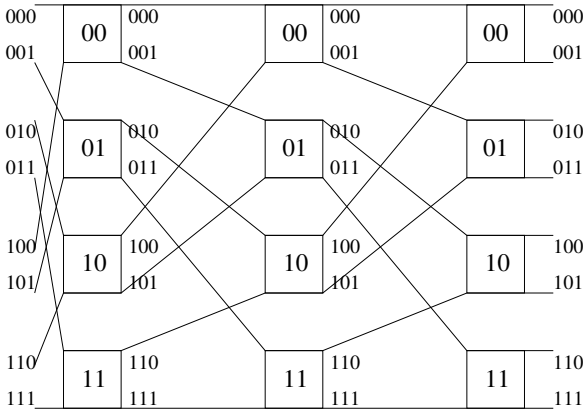


Figure 1.33: Banyan network with Shuffle layout (ex. 41)

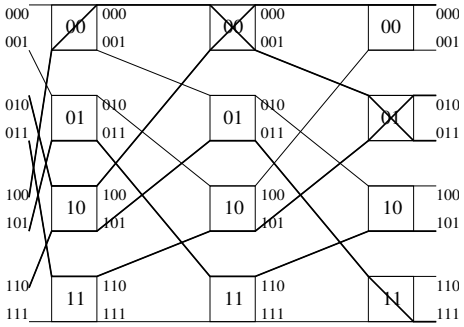


Figure 1.34: Banyan network with Shuffle layout - non-blocking configuration (ex. 41)

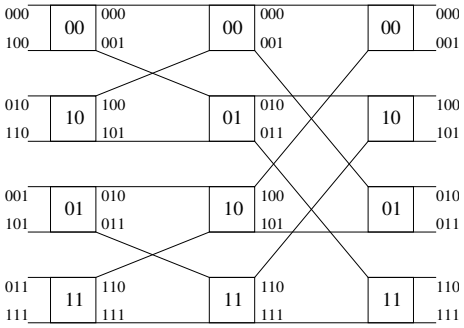


Figure 1.35: Banyan network with Banyan layout (ex. 42)

Exercise 43
Design a self-routing network, of size 8×8*.*

1. *From which other network is possible to obtain the required network? How?*
2. *Draw the network, identify all the inputs, the outputs and the modules.*
3. *Show how the following connections are self-routed:* $1 \to 8$, $2 \to 7$, $3 \to 6$.
4. *Is the network blocking? Why?*

★★★

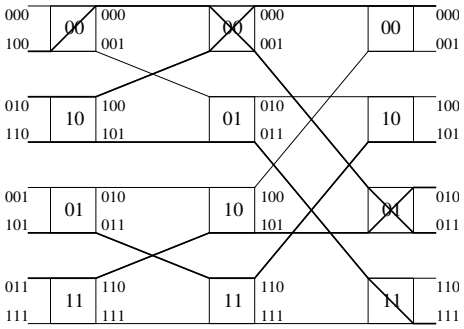


Figure 1.36: Banyan network with Banyan layout - non-blocking configuration (ex. 42)

Exercise 44
Design a Banyan network, of size 16×16*.*

1. *From which other network is possible to obtain the required network? How?*
2. *Draw the whole network, identify all the inputs, the outputs and the modules.*
3. *Banyan networks are blocking. Provide an example of blocking configuration.*
4. *How is it possible to design a complete non-blocking network starting from a Banyan one?*

★★★

Exercise 45
Design a Banyan network, of size 8×8*.*

1. *From which other network is possible to obtain the required network? How?*
2. *Draw the whole network, number properly all the inputs, the outputs and the modules.*
3. *Banyan networks are blocking. Provide an example of blocking configuration with at least 4 connected input-output pairs.*
4. *How is it possible to design a non-blocking network starting from a Banyan one?*

1.4 Cantor networks

Exercise 46 (out of scope for the program 2021/22)
Draw a generic $N \times N$ *Cantor network. What are its properties? Prove formally the network complexity.*

★★★

Exercise 47 (out of scope for the program 2021/22)
Design a 8×8 *Cantor network.*

1. *What are the properties of such network?*
2. *Draw the complete network.*
3. *Describe the algorithm to configure the network.*

4. *Configure the network following exactly the following sequence of input-output couples:* $4 \to 1, 5 \to 6, 6 \to 2, 7 \to 4, 8 \to 3, 1 \to 5, 2 \to 8, 3 \to 7$.

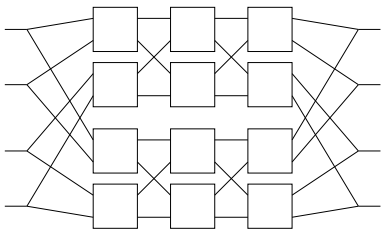
★★★

Exercise 48 (out of scope for the program 2021/22)
Consider a 4×4 *Cantor network.*

1. *Draw it.*
2. *Discuss its properties in terms of blocking.*
3. *Describe the routing algorithm to configure the network.*
4. *Show the connections to support the following input-output couples:* $1 \to 4$, $2 \to 3$, $3 \to 1$ *and* $4 \to 2$.
5. *Apply Lee's method to evaluate the blocking probability of the network. (question out of scope for the program 2021/22)*
6. *Is the blocking probability evaluated by Lee's method compatible with the blocking properties of the network? Why?*
7. *Is there any alternative switching architecture based on* 2×2 *modules, with the same blocking properties of Cantor network, but lower complexity? If yes, which one?*

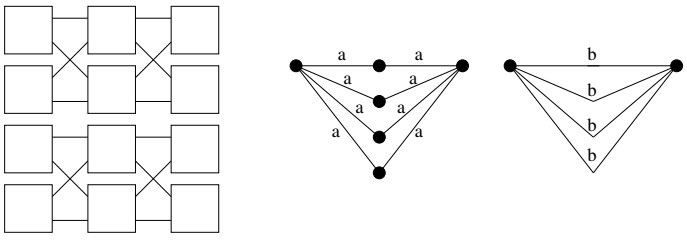
Solution:

The corresponding Cantor network is in the figure.



The algorithm to configure the network is a just a breadth-first search in the corresponding pi-graph, considering only the available links.

The reductions in the pi-graph is shown in the figure, where only the links for which a contention may occur are shown. Note that the edges from the input to each Benes network and from each of them to the outputs are always available.



Let ρ *be the average load at any input. Now:* $a = 4\rho/8 = \rho/2$, $b = 1 - (1 - a)^2$ *and finally the blocking probability is*

$$P(\textit{blocking}) = b^4 = (1 - (1 - \rho/2)^2)^4 = \rho(1 - \rho/4)^4$$

Note that for this simple case it is possible to improve the approximation by using $\rho = 3\rho/8$ since, when adding a new circuit, the average number of pre-existing circuits is 3ρ and not 4ρ as used in evaluating a above.

In both cases, when $\rho = 1$, the blocking probability appears to be larger than 0 (precisely, around 0.15-0.30), even if the network has been designed as strictly non-blocking. This is due to the approximated method adopted.

An example of alternative network is a classical Clos network, built with 2×2 modules. The final complexity will be:

$$C_{4\times4} = 2C_{2\times3} + 3C_{2\times2} + 2C_{3\times2} = 4C_{2\times3} + 3C_{2\times2} = 8C_{2\times2} + 3C_{2\times2} = 11C_{2\times2}$$

which is slightly better (one less module) than the Cantor network proposed above. The control algorithm is the same.

★★★

Exercise 49 (out of scope for the program 2021/22)
Draw a generic $N \times N$ Cantor network.

1. *How can be designed?*
2. *What is its blocking probability?*
3. *Prove formally the network complexity, after defining properly the adopted notation.*
4. *Which algorithm can be used to route the traffic on it?*

1.5 Comparison among networks

Exercise 50 (out of scope for the program 2021/22)
Compute the number of possible switching states *for a crossbar, a Banyan network and a Benes networks, all of them of size $N \times N$. According to this computation, which network is blocking? Why? What is the probability that a random permutation is blocking for a Benes network? If useful, the Stirling approximation is the following, for $n \to \infty$: $\log n! \approx n \log n - n$.*

Solution: The crossbar allows $N!$ configurations, which is equal to the number of switching states; asymptotically, for $N \to \infty$, the number of configurations is $S_{crossbar} = N^N$, using Stirling approximation.

The Benes network is composed by $(2\log_2 N - 1)N/2$ modules of size 2×2, each of them with only two possible states. Hence, the number of switching states of the Benes network is: $2^{(2\log_2 N-1)N/2}$. Asymptotically, this is $2^{(2\log_2 N)N/2} = 2^{N\log_2 N}$, from which $S_{Benes} = N^N$.

The Banyan network is composed by $(\log_2 N)N/2$ modules of size 2×2. Hence, the number of switching states is $2^{(\log_2 N)N/2}$, which is equal, asymptotically, to $S_{Banyan} = N^{N/2}$.

A necessary condition for non-blocking is that the number of states in the switching matrix is at least equal to the number of possible permutations $N! \approx N^N$. This condition holds only for the crossbar and the Benes network, whereas it does not hold for the Banyan network:

$$S_{Banyan} \ll S_{crossbar} \approx S_{Benes}$$

Indeed, the Banyan network is blocking.

Now the probability that a permutation in a Banyan network is non-blocking is:

$$\frac{S_{Banyan}}{N!} \approx \frac{N^{N/2}}{N^N} = e^{\frac{N}{2}\log N - N\log N} = e^{-\frac{N}{2}\log N}$$

Hence the probability of a blocking configuration is:

$$1 - e^{-\frac{N}{2}\log N}$$

which saturates to 1 very quickly when N grows: this means that almost all configurations are blocking.

★★★

Exercise 51 (out of scope for the program 2021/22)

1. *Define formally: (i) Clos network, (ii) Benes network, (iii) Cantor network and (iv) Banyan network.*
2. *Compute the theoretical complexity in terms of contact points, in function of the number N of the ports.*
3. *Which network is the less complex? With which disadvantages?*

Solution:

- Clos network:
  - 3 stages
  - complete interconnection web between stages
  - $C(\text{Clos}) \geq (2p-1)(2pq+q^2)$ if strictly non-blocking; $C(N) \geq 2qp^2+pq^2$ if rearrangeable; where $pq = N$.
- Benes network:
  - Clos network
  - rearrangeable
  - recursively factorized with factor 2
  - $N = 2^n$ and $p = 2$
  - $C(\text{Benes}) = 4N(\log_2 N - 1) + 2N$
- Cantor network:
  - strictly non-blocking
  - recursively factorized with factor 2
  - built by putting $\log_2 N$ Benes networks in parallel, with N de/multiplexer $1 : \log_2 N$ at the first and at the last stage
  - $C(\text{Cantor}) = \log_2 N C(\text{Benes}) \approx 4N(\log_2 N)^2$
- Banyan network
  - blocking network
  - self-routing
  - built by cutting after the $\log_2 N$-th stage in the Benes network
  - $C(\text{Banyan}) = 2N \log_2 N$

The Banyan network, even if less complex, cannot be used alone as switching network since it is blocking.

N	Crossbar	Benes	Clos $\sqrt{N}$	Clos $N/2$
4				
16				
N				

Table 1.3: Complexity in terms of contact points (Ex. 52)

N	Crossbar	Benes	Clos $\sqrt{N}$	Clos $N/2$
4	1 module 4×4	6 mod. 2×2	6 mod. 2×2	6 mod. 2×2
16	1 m. 16×16	56 m. 2×2	12 m. 4×4	16 m. 2×2, 2 m. 8×8
N	1 m. $N \times N$	$N \log N - \frac{N}{2}$ m. 2×2	$3\sqrt{N}$ m. $\sqrt{N} \times \sqrt{N}$	N m. 2×2, 2 m. $\frac{N}{2} \times \frac{N}{2}$

Table 1.4: Features for the networks designed at point 1

N	Crossbar	Benes	Clos $\sqrt{N}$	Clos $N/2$
4	16	24	24	24
16	256	224	192	192
N	N^2	$4N \log N - 2N$	$3\sqrt{N}N$	$4N + N^2/2$

Table 1.5: Complexity in terms of points of contacts

★★★

Exercise 52
Design a non-blocking, rearrangeable network of size $N \times N$, in the cases $N = 4$ and $N = 16$. The considered architectures are: crossbar, Benes network, Clos network with $\sqrt{N}$ modules at the first stage, Clos network with $N/2$ modules at the first stage.

1. *In all cases and with the four architectures, draw the corresponding networks with all interconnections, specifying the number and dimensions of every module.*
2. *Complete Table 1.3.*
3. *In each of the three cases: $N = 4$, $N = 16$, $N \to \infty$, which network minimizes the complexity?*

Solution:

1. We omit the drawing, which should have the features reported in table 1.4.
2. See Table 1.5.
3. In the case $N = 4$, the network with the minimum complexity is the crossbar, for $N = 16$ it is the Clos network (in both cases), for $N \to \infty$ it is the Benes network.

1.6 Lee method

Exercise 53 (out of scope for the program 2021/22)
Consider a 12×16 interconnection network for circuit switching, composed by 9 switches interconnected as a 3 stage network, with double links between each pair of switches in the 1st and 2nd stage, as shown in Fig. 1.37.

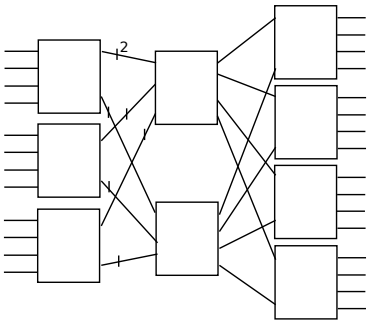


Figure 1.37: Circuit-switching topology (Ex. 53).

1. *Using the Lee method, compute the blocking probability in function of the normalized offered load h, with $h \in [0, 1]$, at each input port. Draw the original graph and show all the steps for its reduction.*
2. *What are the traffic admissibility conditions for the considered network? Provide an intuitive explanation for them.*
3. *Assume that the total offered traffic is 3 Erlang. Write the formula for the corresponding blocking probability.*
4. *Which routing algorithm can be adopted for each new incoming circuit?*

Solution: 1. By applying the Lee method we have the graph reduction steps shown in Fig. 1.38.

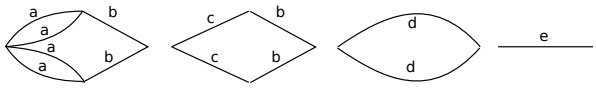


Figure 1.38: Solution for Ex. 53

with

$$c = a^2$$
$$d = 1 - (1 - c)(1 - b) = 1 - (1 - a^2)(1 - b)$$
$$P_b = e = d^2 = (1 - (1 - a^2)(1 - b))^2$$

Given the offered load h at each input:

$$a = 12h/12 = h \qquad b = 12h/8 = 3h/2$$

and thus the blocking probability becomes:

$$P_b = (1 - (1 - h^2)(1 - 3h/2))^2 = (3h/2 + h^2 - 3h^3/2)^2$$

2. The admissibility conditions are

$$a \leq 1, b \leq 1 \qquad \Rightarrow \qquad h \leq 2/3$$

The final condition assures that the links between the 2nd and the 3rd stages are not overloaded, being the “bottleneck” in the architecture.

For any $h > 2/3$, $P_b = 1$.
3. If the total traffic is 3 Erlang, then $h = 3/12 = 1/4$, then

$$P_b = (3/8 + 1/16 - 3/128)^2$$

4. To compute the routing, a simple graph traversal algorithm would work (e.g., depth-first search, breadth-first search) on the graph in which only the available links are considered.

★★★

Exercise 54 (out of scope for the program 2021/22)
Design a rearrangeable Clos network of size 30×30, built with first stage modules with 5 inputs.

1. *Draw the complete network, evaluating the dimensions of every module.*
2. *Compute the total complexity in terms of contact points.*
3. *Using the Lee approximation, compute the blocking probability P_b when ρ, with $\rho < 1$, is the load for each input. Show all the reduction steps on the Lee graph.*
4. *In the case $\rho = 0.5$, compute P_b.*
5. *What does it mean that $P_b > 0$ in this rearrangeable network?*

Solution:

1. First and third stages: 6 modules 5×5. Second stage: 5 modules 6×6.
2. There exist 480 contact points, in total.
3. For the graph reduction, see the class notes. The final formula to obtain is: $P_b = [1-(1-\rho)^2]^k$ where $k = 5$ in this case.
4. Applying the formula for $\rho = 0.5$: $P_b = 0.2373$.
5. The network is rearrangeable, hence it is not blocking. This seems to be in contradiction with $P_b > 0$. But, first, the Lee model is approximated since it does not take into account the correlation among connections. Second, the blocking probability computed by Lee method refers to the fact that, given a connection state in the interconnection network, an idle input cannot be connected to an idle output; in this case, the network is reconfigured. Hence, P_b can be seen as the approximated reconfiguration probability for a rearrangeable network.

★★★

Exercise 55 (out of scope for the program 2021/22)
Design a 512×512 switch, for which the blocking probability is less than 0.001. Each input is observed busy on average for 260 seconds on an observation interval of 320s.

1. *Compute the total traffic (measured in Erlang) loading the switch.*
2. *Design the switch with a Clos network, in which the inputs are divided in 32 modules of size $16 \times k$:*
   - *Draw the final network and specify the dimension of every module.*
   - *Compute k to satisfy the blocking requirement and to minimize the total complexity of the switch.*

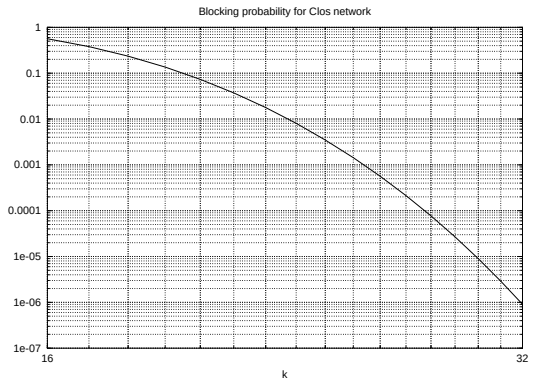


Figure 1.39: Blocking probability for Clos network

- *Compute the final complexity in terms of number of contact points.*
- *What would be the value of k and the complexity if the network was strictly non blocking?*

3. *Design the switch with a symmetric network with two stages, in which inputs are divided among 32 modules.*
   - *Draw the final network and specify the dimension of every module.*
   - *Compute the blocking probability.*
   - *If the blocking requirement is not satisfied, consider the same network with two stages but with l parallel links. Compute l to satisfy the blocking requirement and to minimize the total complexity of the switch.*
   - *Compute the final complexity in terms of number of contact points.*
4. *Design the switch with a crossbar.*
   - *Compute the final complexity in terms of number of contact points.*
5. *Compare the performance and the complexity of the three solutions considered above.*

Solution:

1. $$\rho_{tot} = 260/320 \times 512 = 416 \text{ Erlang}$$
2. Clos network: Through the graph reduction, it is obtained:
   $$a = P(\text{busy link}) = \frac{\rho_{tot}}{32k}$$
   from which the blocking probability is:
   $$P_b = (1-(1-a)^2)^k = \left(1-\left(1-\frac{\rho_{tot}}{32k}\right)^2\right)^k$$
   which is shown in Fig. 1.39. Hence, $k = 26$ meets the blocking requirement. The final complexity is:
   $$C = 2 \times 32 \times (16 \times 26) + 26 \times (32 \times 32) = 53248$$

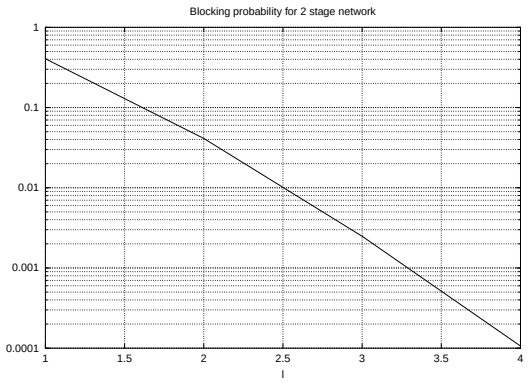


Figure 1.40: Blocking probability for the two stage switch

If the network was strictly non blocking, then $k = 31$ and

$$C = 2 \times 32 \times (16 \times 31) + 31 \times (32 \times 32) = 63488$$

3. Two stages network: 32 modules are present at the first stage, of size 16×32. The second stage is symmetric to the first one. The blocking probability is:

   $$P_b = \rho_{tot}/32^2 = 0.41$$

   If there are l parallel links, the blocking probability becomes:

   $$P_b = \left(\frac{\rho_{tot}}{1024l}\right)^l$$

   which is shown in Fig. 1.40. To obtain the required P_b, it is enough to set $l = 4$. The number of corresponding contact points is:

   $$C = 2 \times 32(16 \times 32) \times 4 = 131072$$

4. Crossbar: The blocking probability is zero. The final complexity is:

   $$C = 512^2 = 262144$$

★★★

Exercise 56 (out of scope for the program 2021/22)
Consider a 16×16 Clos network, symmetric, rearrangeable, with (case A) 2 inputs for each first-stage module, (case B) 4 inputs for each first-stage module.

Draw the corresponding pi-graph for both cases and compute the blocking probability. Is the final network blocking? In both cases? Why? What is the meaning of this result?

★★★

Exercise 57 (out of scope for the program 2021/22)
Design an 8×4 switch (i.e., a concentrator) using a Clos network in which the number of modules at the i-th stage is r_i. In both the following cases, design the network such that it is rearrangeable non blocking and with minimum complexity.

- **Case A***: $r_1 = 4$ and $r_3 = 2$.*
- **Case B***: $r_1 = 4$ and $r_3 = 1$.*

In both cases:

1. *Draw the total network, with all modules and all interconnections.*
2. *Compute the total complexity.*
3. *Draw the Paul matrix and explain the meaning of each row, each column and each element of the matrix.*
4. *What is the algorithm to configure the network?*
5. *Using the Lee method, compute the blocking probability when the* total *offered load to the concentrator is 2 Erlang.*
6. *Is the blocking probability non null? Why? What is the meaning of such result?*
7. *According to Lee method, which case is with the lower blocking probability? Does it correspond to the case with higher or lower complexity?*

Solution:

- Questions 1, 2, 5:

Case A.

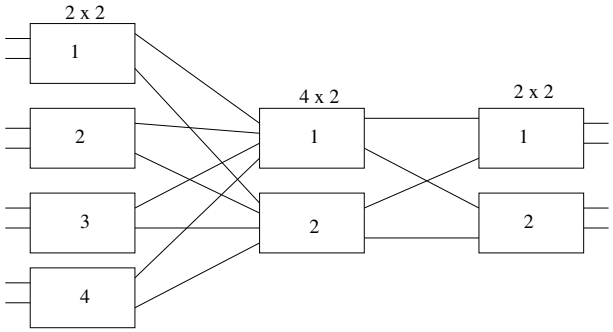


Figure 1.41: Rearrangeable Clos network in case A.

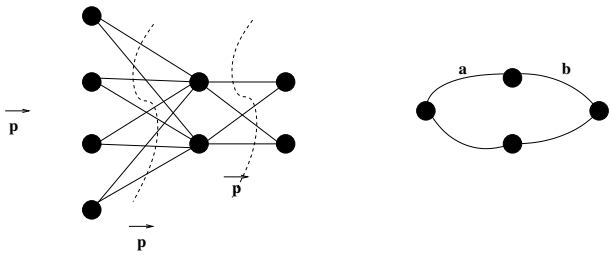


Figure 1.42: Equivalent graph for Clos network in case A.

$$C(8 \times 4) = 4C(2 \times 2) + 2C(4 \times 2) + 2C(2 \times 2) = 40$$

$$p = 2\ Erlang,\ \ a = \frac{p}{4 \times 2} = 0.25,\ \ b = \frac{p}{2 \times 2} = 0.5$$

$$P(\text{block}) = [1 - P(\text{free})]^2 = [1 - (1-a)(1-b)]^2 \simeq 0.39$$

Case B.

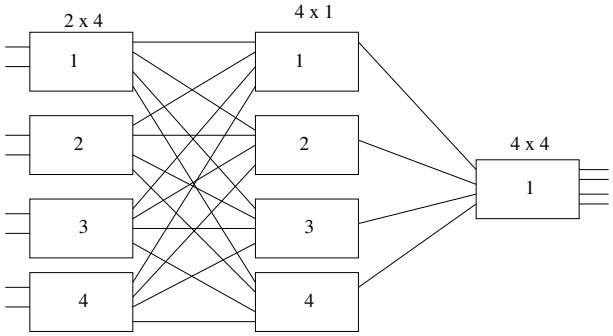


Figure 1.43: Rearrangeable Clos network in case B.

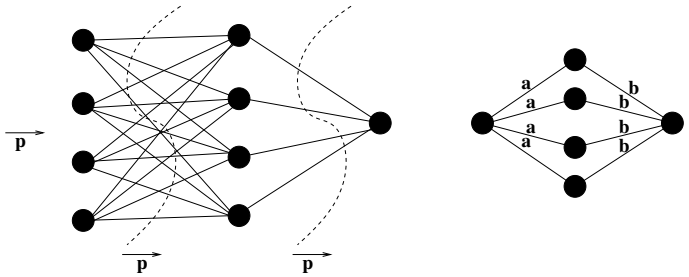


Figure 1.44: Equivalent graph for Clos network in case B.

$$C(8 \times 4) = 4C(2 \times 4) + 4C(4 \times 1) + C(4 \times 4) = 64$$

$$p = 2\ Erlang,\ \ a = \frac{p}{4 \times 4} = 0.125,\ \ b = \frac{p}{4} = 0.5$$

$$P(\text{block}) = [1 - P(\text{free})] = [1 - (1-a)(1-b)]^4 \simeq 0.10$$

- Questions 3, 4: see class notes.
- Question 6: The blocking probability is non null, even if the network is non blocking, because of the approximations of Lee method (uniform traffic and uncorrelated state of the links among different stages). This probability can be thought as the probability of rearrange the network.
- Question 7: case B is the one with lower blocking probability, but it corresponds to an higher complexity.

★★★

Exercise 58 (out of scope for the program 2021/22)
Design an 16×4 switch (i.e., a concentrator) using a Clos network in which the number of modules at the i-th stage is r_i. In both the following cases, design the network such that it is rearrangeable non blocking and with minimum complexity.

- **Case A**: $r_1 = 4$ *and* $r_3 = 2$.
- **Case B**: $r_1 = 4$ *and* $r_3 = 1$.

In both cases:

1. *Draw the total network, with all modules and all interconnections.*
2. *Compute the total complexity.*
3. *Draw the Paul matrix and explain the meaning of each row, each column and each element of the matrix.*
4. *What is the algorithm to configure the network?*
5. *Using the Lee method, compute the blocking probability when the* total *offered load to the concentrator is 2 Erlang.*
6. *Is the blocking probability non null? Why? What is the meaning of such result?*
7. *According to Lee method, which case is with the lower blocking probability? Does it correspond to the case with higher or lower complexity?*

★★★

Exercise 59
Design an 8×8 *Benes network.*

1. *Draw the complete network.*
2. *Which algorithms can be used to configure the network?*
3. *Connect: 1-5, 3-2, 4-3, 5-4, 8-1.*
4. *Compute the blocking probability according to the Lee method,* $P_b(\rho)$*, as function of* ρ*, the average single-input load. (question out of scope for the program 2021/22)*
5. *What is the meaning of* $P_b(\rho) > 0$ *for this network? (question out of scope for the program 2021/22)*
6. *In the case* $\rho = \frac{k}{8}$*, for* $k = 0, \ldots, 8$*, what is the meaning of* $P_b(k)$*? (question out of scope for the program 2021/22)*

Solution: The algorithm to configure is either the looping algorithm or the Paull's algorithm, both applied recursively in the network. Looping algorithm, derived from Paull's algorithm, is based on the full knowledge of all the input-output connections before the switching network is configured. On the contrary, Paull algorithm is incremental and adds just one input-output connection at the time, and rearranges the network when needed.

Fig. 1.45 reports the pi-graph for the Benes network, with all the reduction steps. The loads for the edges are the following: $a = \rho$, $b = 1 - (1-\rho)^2$, $c = b^2 = (1-(1-\rho)^2)^2$, $d = 1 - (1-\rho)^2[1-(1-(1-\rho)^2)^2]$ and finally $P_b(\rho) = d^2$:

$$P_b(\rho) = \left(1 - (1-\rho)^2[1-(1-(1-\rho)^2)^2]\right)^2$$

This $P_b(\rho)$ represents the approximated probability that the network is rearranged in the case a new connection is added, with ρ the probability of busy for a single port. When $\rho = k/N$, on average there are k active inputs and $P_b(k)$ represents the approximated probability that a network with already k connections is rearranged when a new connection between an idle input and idle output is setup. Note that this probability refers only to Paull's algorithm, which is incremental.

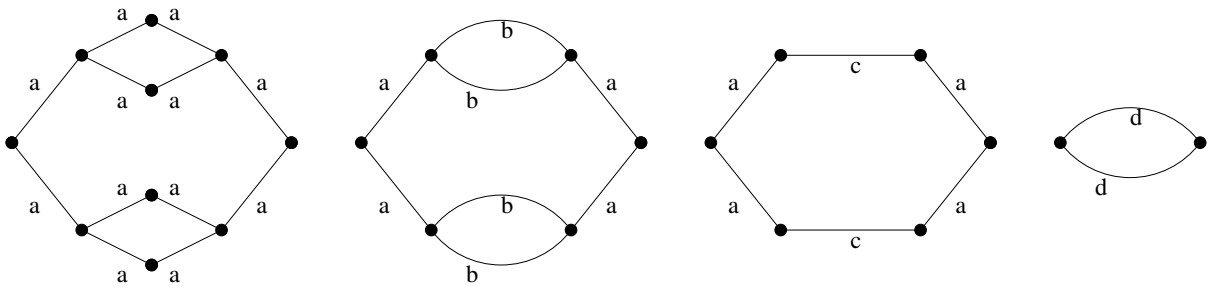


Figure 1.45: Pi-graphs for the Benes network after each reduction.

1.7 Space-time switching

Exercise 60 (out of scope for the program 2021/22)
Design a space-time switch with 3 ports, each of them receiving an E1 frame (32 slots of one byte each, every 125 μs). Plot the STS network and transform it into an SSS network.

★★★

Exercise 61 (out of scope for the program 2021/22)
Consider a space-time switch, in two possible configurations: ST (space-time) and TS (time-space). Which configuration has lower blocking probability? Why?

Solution:

$$P(\text{blocking}_{TS}) < P(\text{blocking}_{ST})$$

See the class notes. The TS switch allows to rearrange the slots and reduce the blocking.

★★★

Exercise 62 (out of scope for the program 2021/22)

1. *Draw a TST network, with N ports and supporting frames of k timeslots.*
2. *Show the equivalent completely spatial network.*
3. *What is the meaning of such equivalence?*

★★★

Exercise 63 (out of scope for the program 2021/22)
Consider a 6×6 Time-Space-Time (TST) switch for digital telephony. Incoming frames comprise 4 timeslots.

1. *Describe the algorithm to configure the timeslot interchangers and the space switch, given a switching configuration to obtain.*
2. *Is the TST switch blocking? Why?*
3. *Use the Lee method to estimate the blocking probability.*

Solution: The TST switch is equivalent to a SSS which is a Clos network, with first/third stage modules 4×4 and second stage modules 6×6; this network is non-blocking and can be configured through Paull algorithm.

When, in the first/third stage module at input/output k of the Clos network, input i connects output j, in the corresponding TSI at input/output k the data from timeslot i is moved to timeslot j. When in the second stage module k of the Clos network input i connects to output j, the corresponding spatial switch of STS moves the data from input i to output j during timeslot k.

★★★

Exercise 64 (out of scope for the program 2021/22)
Consider a 4×4 STS switch built with k TSI and fed by traffic organized in frames of 2 slots.

1. *Draw the STS switch*
2. *Draw the equivalent SSS switch*
3. *Compute the blocking probability of the switch for different values of k. Describe the assumptions made.*

Chapter 2

Data centers

2.1 Cloud computing

Exercise 65
Answer to the following questions related to cloud computing and data centers.

1. *What is cloud computing according to NIST definition?*
2. *For each cloud computing service model, define it and provide an example.*
3. *What is the concept of resource virtualization?*
4. *List the resources that are typically virtualized, and explain what practically mean that they are virtualized.*
5. *Describe the traffic within a data center, specifying the end-points and providing some example.*

★★★

Exercise 66
Answer to the following questions related to cloud computing and data centers.

1. *What are the key aspects of cloud computing, according to the NIST definition?*
2. *Define each cloud computing service model and provide at least two examples.*
3. *What does it mean that the cpu is virtualized in a data center?*
4. *What does it mean that the memory is virtualized in a data center?*
5. *What does it mean that the storage is virtualized in a data center?*
6. *What does it mean that the network is virtualized in a data center?*
7. *Describe the traffic within a data center, specifying the end-points and providing some examples.*
8. *What is the purpose of BGP within a data center?*

2.2 Design of data center networks

Exercise 67
Consider the design of a data center.

1. *What are the main advantages and disadvantages of managing the addressing at level 2 and at level 3?*
2. *Explain the difference between End-of-Row (EOR) and Top-of-Rack (TOR) architectures, highlighting the corresponding advantages and disadvantages.*
3. *What is the purpose of the technology denoted as "multi-chassing link aggregation"?*
4. *What is the difference between overlay and underlay in a data center?*
5. *What is the motivation to use an overlay in a data center?*
6. *What is the role of BGP in a data center?*

★★★

Exercise 68
Consider a data center built with the switches in the table. All the servers are equipped with ports at 10 Gbps.

Switch model	*Ports*
Top-of-Rack	*40 @ 10 Gbps plus 4 @ 40 Gbps*
Spine	*8 @ 40 Gbps*

1. *Draw the largest leaf-and-spine data center with the above switches*
2. *Compute the corresponding number of servers, ToR switches and spine switches*
3. *Compute the corresponding oversubscription ratio*
4. *Assume eBGP used for routing: show a feasible AS number association for each switch*
5. *Assume iBGP used for routing: show a feasible AS number association for each switch*

Solution:

- 8 ToR + 4 spines. 320 servers.
- oversubscription ratio $40 \times 10/(40 \times 4) = 2.5$
- for eBGP, each switch is seen as an AS. Now AS number 101,102,..., 108 for the ToR switches and 201, 202, 203, 204 for the spine switches.
- for iBGP, one AS is shared across all the switches. Just use AS number 101 for all the switches.

★★★

Exercise 69
Consider the design of a data center.

1. *Draw the standard topology used to interconnect the servers and the switches, distributed across many racks. Highlight the role of the ToR (Top-of-Rack) switches.*

2. *What is a POD? How does affect the interconnection topology?*

Consider the specific scenario in which (i) 1024 servers are interconnected; (ii) each rack is equipped with at most 32 servers; (iii) the following switches are available, where x is some given price:

Model	*Number of ports*	*Port rate*	*Cost (€)*
M1	*32*	*1 Gbit/s*	$0.7x$
M2	*64*	*1 Gbit/s*	$1.0x$
M3	*128*	*1 Gbit/s*	$1.5x$

1. *Design the whole interconnection network to implement the given scenario at the minimum cost.*
2. *Evaluate the final cost.*

Solution:

In the first part of the problem it was required to describe the leaf-and-spine topology, highlighting the roles of the racks , of ToR switches and of PODs.

Let C_{TOT} be the total cost. Let $C_{y\ p}$ be cost of a switch with y ports. Three possible design solutions must be considered:

- leaf: 64p, spine: 32p.

$$C'_{TOT} = 32C_{64p} + 32C_{32p} = 32x + 32 \cdot 0.7x$$

- leaf: 64p, spine: 64p.

$$C''_{TOT} = 32C_{64p} + 16C_{64p} = 32x + 16x = 48x$$

- leaf: 64p, spine: 128p.

$$C'''_{TOT} = 32C_{64p} + 8C_{128p} = 32x + 8 \cdot 1.5x = 44x$$

Since $C'_{TOT} > C''_{TOT} > C'''_{TOT}$, the third solution is the cheapest one.

★★★

Exercise 70

Consider the design of a data center based on a leaf-spine topology.

- *Describe the correspondence between the topology and a Clos network, highlighting similarities and differences.*

Assume now a total number of servers equal to 32,000 with each server interface running at 1 Gbit/s. Each rack can host 32 servers and racks are grouped into pods. Each switch is equipped with 64 ports running at 1 Gbit/s.

- *Design the interconnection topology.*
- *Compute the total number of switches and cables required to build the data center.*
- *How many pods result from the design?*
- *Which routing protocols can be used to find the path between two servers? Motivate your answer.*

★★★

Solution:

The complete solution is not reported.

To design a data center with 32,768 servers, there are needed 2,560 switches with 64 ports and 66,560 cables. The total number of pods is 32.

★★★

Exercise 71

Consider the design of a large data center based only on Ethernet switches with P ports.

1. *Describe the advantages and disadvantages to adopt a layer-2 addressing and routing scheme.*
2. *Describe the advantages and disadvantages to adopt a layer-3 addressing and routing scheme.*
3. *For each of the following design problems, draw at the high-level the network, compute the required number of P-port switches and the maximum number of supported servers,*
   (a) *Design the largest possible two level (leaf-and-spine) data center.*
   (b) *Design the largest possible two level (leaf-and-spine) POD.*
   (c) *Design the largest possible three level (POD-and-spine) data center.*
4. *Consider a data center with 400 servers each of them equipped with one port at 1 Gbps. Each rack hosts 40 servers. All the adopted switches have $P = 80$ ports running at 1 Gbps each.*
   (a) *Draw the network.*
   (b) *Compute the number of required 80-port switches.*

Solution:

3.(a) The design uses P switches in the leaf level and $P/2$ switches in the spine level. Thus, the number of servers is $P^2/2$ and the number of switches is $1.5 \times P$.

3.(b) The design uses $P/2$ switches in the leaf level of the POD and $P/2$ switches in the spine level of the POD. Thus, the total number of server ports is $P^4/4$, the number of ports to the spine is $P^2/4$, and the number of switches is P.

3.(c) The design uses P PODs in the leaf level and $P^2/4$ switches in the spine. Thus, the total number of servers is $P^3/4$ and the number of switches is $(5/4) \times P^2$.

4.(a) The leaf level is based on 10 switches whereas the spine level is based on 5 switches. The interconnection is full between the leaf and spine levels, with a parallelism of 8 Gbps.

4.(b) The total number of switches is 15.

★★★

Exercise 72

Consider the design of a massively large data center based only on Ethernet switches, assuming an oversubscription ratio $4:1$ between server capacity and network capacity. All the adopted switches are equipped with 50 ports running at 1 Gbps. Each server is equipped with one port running at 1 Gbps.

1. *For each of the following design problems, draw at high-level the network, compute the required number of switches and the maximum number of supported servers:*

(a) *Design the largest possible 2-levels (leaf-and-spine) data center.*

(b) *Design the largest possible 2-levels (leaf-and-spine) POD.*

(c) *Design the largest possible 3-levels (POD-and-spine) data center.*

(d) *Design the largest possible 3-levels (POD-and-spine) POD.*

(e) *Design the largest possible 4-levels (POD-and-spine) data center.*

2. *Discuss how this design is similar to Google's Jupiter data centers.*

3. *Describe the advantages and disadvantages to adopt a layer-2 addressing and routing scheme if adopted for the above 4-levels data center.*

Solution:

Regarding question 1, see Table 2.1.

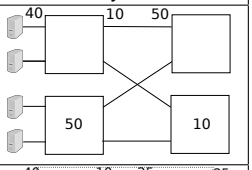
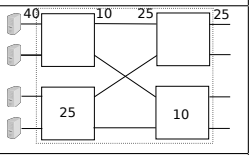
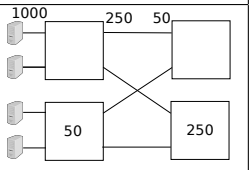
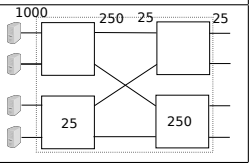
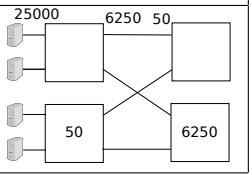
Network	Servers	Spine ports	Switches	Layout
2-levels data center	$40 \times 50 = 2,000$	-	$50 + 10 = 60$	
2-levels POD	$40 \times 25 = 1,000$	$25 \times 10 = 250$	$10 + 25 = 35$	
3-levels data center	$1000 \times 50 = 50,000$	-	$50 \times 35 + 250 = 2,000$	
3-levels POD	$1000 \times 25 = 25,000$	$25 \times 250 = 6,250$	$25 \times 35 + 250 = 1,125$	
4-levels data center	$25,000 \times 50 = 1,250,000$	-	$50 \times 1,125 + 6250 = 62,500$	

Table 2.1: Solution for question 1 of Ex. 72

★★★

Exercise 73

Consider the design of a data center based on switches that can configured as follows:

- *C1: 2 ports at 10 Gbps and 40 ports at 1 Gbps*
- *C2: 6 ports at 10 Gbps*

The design assumes an oversubscription ratio $2 : 1$ *between server capacity and network capacity. Each server is equipped with one port at 1 Gbps.*

1. *For each of the following design problems, draw at high-level the network, compute the required number of switches configured as C1 and configured as C2, and the maximum number of supported servers:*
   (a) *Design the largest possible 2-levels (leaf-and-spine) data center.*
   (b) *Design the largest possible 2-levels (leaf-and-spine) POD.*
   (c) *Design the largest possible 3-levels (POD-and-spine) data center.*
2. *Describe the advantages and disadvantages to adopt a layer-2 addressing and routing scheme if adopted for the above POD-and-spine data center.*

Solution:
As partial solution, the design of the requested data centers is described in the following table.

Design	Servers	C1 switches	C2 switches
2-layers data center	240	6	2
2-layers POD	120	3	2
3-layers data center	720	18	18

Note that 6 PODs are integrated for the last design.

★★★

Exercise 74
Consider the design of a data center based on a leaf-spine topology, with an oversubscription 3:1, connecting 24,576 servers with each server interface running at 10 Gbit/s. Each switch is equipped with 64 ports running at 10 Gbit/s.

1. *Design the interconnection topology, showing all the steps in the design construction.*
2. *Compute the total number of switches and cables required to build the data center.*

Solution: The largest pod that can designed connects 1536 servers to 512 spine switches and it is built with $32 + 16 = 48$ switches. To connects all the servers, $24576/1536 = 16$ pods are necessary. Now the total number of leave switches connecting the pods is 128, with 4 parallel links connected to each pod.
Thus, the total number of switches is $16 \times 48 + 128 = 896$. The total number of cables is

$$24576 + 16 \times 32 \times 16 + 512 \times 16 = 24576 \times (1 + 2/3) = 40960$$

★★★

Exercise 75
Consider the design of a data center based on a leaf-spine topology, with an oversubscription equal to 2:1, with each server interface running at 10 Gbit/s. Each switch is equipped with 60 ports running at 10 Gbit/s.

1. *Design the largest possible two level (leaf-and-spine) data center.*
2. *Design the largest possible two level (leaf-and-spine) POD and connect 6 of them to build the largest 3 levels data center.*

For each of the two data centers, show the interconnection topology and compute (i) the total number of supported servers, (ii) the total number of switches, (iii) the total number of required cables (including the ones required to connect the servers).

Solution:

Data center	Number of servers	Number of switches	Number of cables
2 layers	$40 \times 60 = 2400$	$60 + 20 = 80$	$2400 + 20 \times 60 = 3600$
3 layers	$1200 \times 6 = 7200$	$(30 + 20) \times 6 + 60 = 360$	$7200 + (30 \times 20 \times 6) + 3600 = 14400$

★★★

Exercise 76
Design of a data center, with an oversubscription equal to 4:1, built only with switches with 100 ports.

1. *Design the largest possible two layers data center (i.e., leaf-spine).*
2. *Design the largest possible three layers data center.*

For each of the two data centers, show the interconnection topology and compute (i) the total number of supported servers, (ii) the total number of switches, (iii) the total number of required cables (including the ones required to connect the servers).

Solution:

Topology	Servers	Switches	Cables
2 layer data center	$80 \times 100 = 8000$	$100 + 20 = 120$	$8000 + 20 \times 100 = 10000$
2 layer POD	$80 \times 50 = 4000$	$50 + 20 = 70$	$4000 + 20 \times 50 = 5000$
3 layer data center	$4000 \times 100 = 400000$	$70 \times 100 + 1000 = 8000$	$100 \times 5000 + 1000 \times 100 = 600000$

★★★

Exercise 77
Consider the design of Jupiter data center at Google, based on a basic building block implemented with a chipset with 16 ports at 40 Gbps. The adopted oversubscription ratio is 3:1. Each server is equipped with a single port running at 10 Gbps. Draw the architecture and compute the total number of servers and basic building blocks (i.e., chipsets) for each of the following scenarios:

1. *2-layers topology;*
2. *2-layers POD;*
3. *3-layers topology;*
4. *3-layers POD;*
5. *4-layers topology.*

Finally, describe for the 4-layers topology all the possible ways to interconnect the data center to the Internet.

Solution: See the class notes on Jupiter data center.

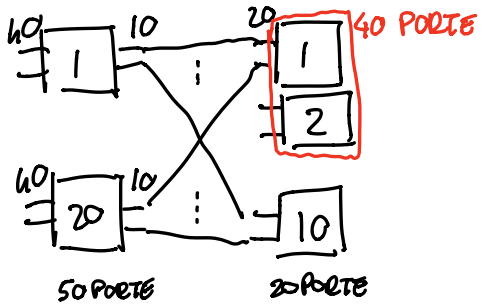
★★★

Exercise 78
Consider the topology of a data center with 2 layers, built only with switches with 50 ports running at 1 Gbps, interconnecting 800 blade servers, each with a single port at 1 Gbps. The adopted oversubscription ratio is 4:1. The rack is a standard 19-inch one. Assume that the average total traffic generated at each server interface is λ Gbps, with $\lambda \in [0, 1]$.

1. *Design the overall interconnection network connecting the servers.*
2. *Compute the required number of switches and racks.*
3. *Draw the corresponding Clos network*
4. *Apply the Lee method to evaluate the blocking probability on the Clos network in function of λ. (question out of scope for the program 2021/22)*
5. *How it is possible to interpret the blocking probability for the original data center scenario? Under which assumptions? (question out of scope for the program 2021/22)*

Solution:

1. The topology is shown in figure.

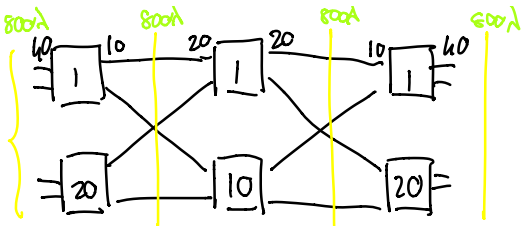


2. Let S_x be a switch with x ports. The total number of switches is:

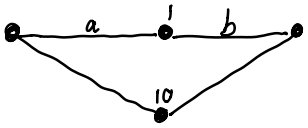
$$20S_{50} + 10S_{20} = 20S_{50} + 5S_{50} = 25S_{50}$$

The required number of racks is 20 for the leaf layer and 1 for the spine layer, thus a total of 21 racks are needed.

3. The Clos network is shown in the figure:



4. To apply the Lee method, the pi-graph is shown below:



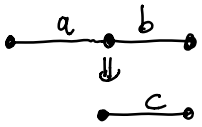
Let ρ be the normalized load on a link defined as:

$$\rho = \frac{\lambda[\text{Gbps}]}{1[\text{Gbps}]}$$

Now:

$$a = b = \frac{800\rho}{10 \times 20} = 4\rho$$

Note that it must $\rho \leq 0.25$ to ensure admissible traffic. The first reduction is shown below:



where

$$c = 1 - (1 - a)(1 - b) = 1 - (1 - 4\rho)^2 = 8\rho - 16\rho^2$$

The second reduction is shown below:

where

$$d = c^{10} = (8\rho - 16\rho^2)^{10}$$

In summary, the total blocking probability for the Clos network is:

$$P_b = (8\rho - 16\rho^2)^{10}$$

with $\rho \leq 0.25$ (i.e., $\lambda \leq 250$ Mbps).

5. The block event in the Clos network corresponds to the event that all the paths from an idle input to an idle output are busy. Thus, in the data center network, this event corresponds to fact that all paths from an idle ToR switch to another idle ToR switch are "busy", i.e. congested by other traffic flows. Thus, P_b can be seen as the level of congestion in the data center. And $(1 - P_b)$ can be seen as the probability that a new flow from an idle ToR switch to another idle ToR switch will experience very small latency, due to the lack of congested links along the path.

   The Lee method is an approximated method and implies strong assumption on the data center system, limiting its applicability: (1) the traffic is uniformly distributed between any pair of ToR switches, (2) each server is source/destination of at most one traffic flow, (3) the routing algorithm distributes randomly the traffic across the topology and all the paths between two ToR traverse always the spine layer, (4) the congestion state among links is independent for all the data center links.

★★★

Exercise 79

Consider the design of a massively large data center based only on Ethernet switches, assuming an oversubscription ratio $3 : 1$ between server capacity and network capacity. All the adopted switches are equipped with 40 ports running at 10 Gbps. Each server is equipped with one port running at 10 Gbps.

1. *For each of the following design problems, draw at high-level the network, compute the required number of switches and the maximum number of supported servers:*

   (a) Design the largest possible 2-levels (leaf-and-spine) data center.

   (b) Design the largest possible 2-levels (leaf-and-spine) POD.

   (c) Design the largest possible 3-levels (POD-and-spine) data center.

(d) *Design the largest possible 3-levels (POD-and-spine) POD.*

(e) *Design the largest possible 4-levels (POD-and-spine) data center.*

2. *Based on above results, design a data center connecting 120,000 servers.*

3. *As summary, fill the following:*

Network	*Max num. servers*	*Num. spine ports*	*Tot. num. switches*
2-levels data center			
2-levels POD			
3-levels data center			
3-levels POD			
4-levels data center			
-	$120,000$		

4. *Discuss how this design is similar to Google's Jupiter data centers.*

Solution:

Regarding question 3:

Network	Max num. servers	Num. spine ports	Tot. num. switches
2-levels data center	$30 \times 40 = 1,200$	-	$40 + 10 = 50$
2-levels POD	$30 \times 20 = 600$	$20 \times 10 = 200$	$20 + 10 = 30$
3-levels data center	$600 \times 40 = 24,000$	-	$40 \times 30 + 200 = 1,400$
3-levels POD	$600 \times 20 = 12,000$	$20 \times 200 = 4,000$	$20 \times 30 + 200 = 800$
4-levels data center	$12,000 \times 40 = 480,000$	-	$40 \times 800 + 4,000 = 36,000$
-	$12,000 \times 10 = 120,000$	-	$10 \times 800 + 1,000 = 9,000$

The last case is obtained by exploiting the 4-levels data center and aggregating 4 spine switches (with 30 unused ports each) into a single one.

★★★

Exercise 80

Consider a modular data center interconnection based on a leaf-and-spine topology, built using switches with 128 *ports running at 1 Gbit/s.*

1. *Design a data center connecting 2048 servers.*

   (a) *describe the equivalent Clos network adopted for the design (question out of scope for the program 2021/22)*

   (b) *describe the final topology*

   (c) *compute the total number of switches*

2. *Extend the above design to implement a POD with 2,048 server ports and 2,048 switch ports (i.e., used to connect to the spine)*

   (a) *describe the final topology*

   (b) *compute the total number of switches*

3. *Connect 8 PODs with the above architecture to design a data center with 16,384 servers*

   (a) *describe the equivalent Clos network adopted for the design (question out of scope for the program 2021/22)*

   (b) *describe the final topology*

   (c) *compute the total number of switches*

4. *Answer to the following questions:*

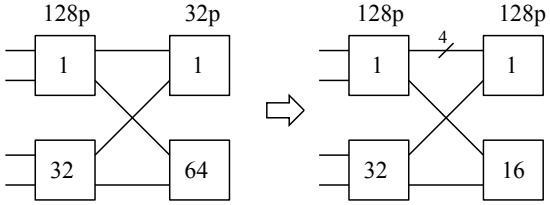
   (a) Define the cloud computing services that can run in a data center

   (b) What is a POD?

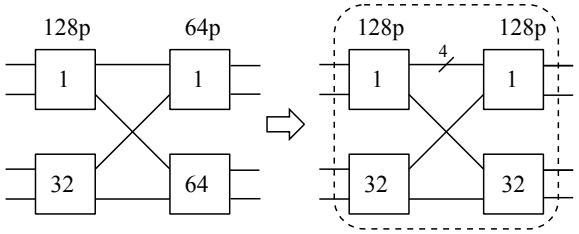
Solution:

For simplicity, we do not report the drawing for the corresponding Clos networks.

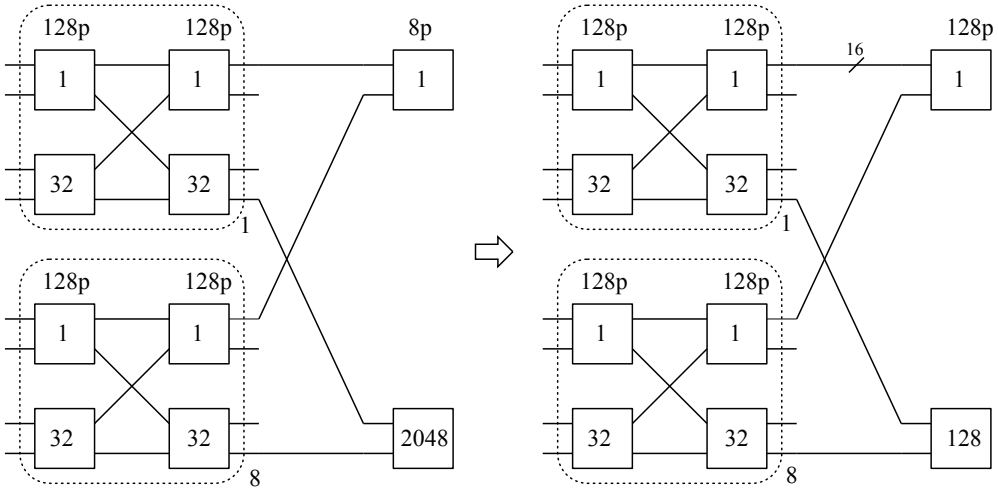
1. The data center with 2,048 servers can be built considering a rearrangeable Clos network with 2,048 ports and 64 ports at the first/third stage module. Thus, it requires 32 switches with 128 ports (leaf layer) and 64 switches with 32 ports (spine layer). Note now that a group of 4 switches with 32 ports in the leaf layer is built with one switch with 128 ports. Thus, the final number of switches with 128 ports is $32 + 64/4 = 48$.



2. To build a POD, the number of switches of the original leaf layer (in left topology of the above figure) does not change but now each switch in the POD spine must have 64 ports to connect to the new spine layer interconnecting the PODs. Thus, the total number of switches with 128 ports in the POD spine is $64/2 = 32$. The complete POD can be built with $32 + 32 = 64$ switches with 128 ports.



3. Finally, the data center with 16,384 servers can be built considering a rearrangeable Clos network with 16,384 ports and 2,048 ports at the first/third stage module. Thus, it requires 8 PODs with 4,096 ports and, for the new spine, 2048 switches with 8 ports. Note that a group of 16 switches with 8 ports is implemented with one switch with 128 ports. Thus, the final number of switches with 128 ports in the spine is 128. In the complete data center, the total number of switches with 128 ports is $8 \times 64 + 128 = 640$.



★★★

Exercise 81
Consider a modular data center interconnection based on a leaf-and-spine topology, built using switches with 128 *ports running at 1 Gbit/s. Assume that each rack is able to host either 64 physical servers and one switch with 128 ports, or 32 switches alone.*

Design a massively large data center with $262144 = 2^{18}$ *physical servers.*

1. *Show the topology of the equivalent Clos network at each level of factorization (question out of scope for the program 2021/22)*
2. *Show the topology interconnecting the 128 port switches*
3. *How many switches and racks are needed in total?*

Solution: Let C_y be a unidirectional switch of size $y \times y$. Let C_{yp} be a switch with y bidirectional ports. Two possible strategies for the design can be followed.

Approach 1. As preliminary step, we can design a 4096×4096 Clos network (since $4096 = (128/2)^2$) as follows:

$$C_{4096} = 2 \times 64C_{64} + 64C_{64}$$

Then we can design a 262144×262144 Clos network based on the above network as follows:

$$C_{262144} = 64 \times 2C_{4096} + 4096C_{64}$$

Now we can use the above Clos topology to build the data center with 128 port switches. A pod with 8192 ports can be built with

$$C_{8192p} = 64C_{128p} + 64C_{128p} = 128C_{128p}$$

requiring 64 racks (for servers and ToR switches) and 2 racks (for spine switches), thus in total 66 racks. Thus, the final data center can be built with

$$\begin{aligned} C_{262144p} = 64C_{8192p} + 4096C_{64p} = 64 \times 128C_{128} + (4096/2)C_{128p} \\ = 8192 + 2048C_{128p} = 10240C_{128p} \end{aligned}$$

requiring 64×66 racks for the pods and $2048/32 = 64$ racks for the spine, thus in total 4288 racks.

Approach 2. We design a 262144×262144 Clos network as follows:

$$C_{262144} = 2 \times 4096C_{64} + 64C_{4096}$$

in which we have:

$$C_{4096} = 2 \times 64C_{64} + 64C_{64}$$

Now we can use the above Clos topology to devise the overall data center:

$$C_{262144p} = 4096C_{128p} + 64C_{4096p}$$

which requires 4096 racks (for servers and ToR switches) and some additional ones to host the pods. Now each 4096 port pod can be built as

$$C_{4096p} = 64C_{128p} + 64C_{64p} = (64 + 32)C_{128p} = 96C_{128p}$$

which requires $96/32 = 3$ racks. Thus, the overall number of 128 port switches is $4096 + 64 \times 96 = 10240$ (as in approach 1) and the overall number of racks is $4096 + 64 \times 3 = 4288$ (as in approach 1).

Chapter 3

Packet switches

3.1 Theoretical performance

3.1.1 Bufferless switches

Exercise 82 (out of scope for the program 2021/22)
Consider a generic $N \times N$ switch, without buffers, with synchronous operation: at most one packet is transferred for each timeslot from each input and to each output. The traffic is uniform and p is the arrival probability of a packet at an input during one timeslot. When many packets arrive destined for the same output, just one of them is transferred to the output whereas the others are lost.

- *What is the average number of lost packets for each timeslot?*
- *Compute the average number of lost packets in the case $N = 8, 16, 256$ e $p = 0.1, 0.5, 1.0$.*
- *When the packets arrive at 1 Gbps, compute the maximum speed at which packets leave from each port, in the cases $N = 8, 16, 256$.*
- *Compute the limiting throughput when $N \to \infty$.*

Solution: Let X be a random variable equal to the number of packets arrived and destined to output U. An input link does not send any packet with probability $1 - p$, sends a packet to U with probability p/N and sends to another output with probability $(N - 1)p/N$. Hence the probability that, given N inputs, there exist x packets destined to U is the following:

$$P(X = x) = \binom{N}{x} \left(\frac{p}{N}\right)^x \left(1 - \frac{p}{N}\right)^{N-x}$$

for $x = 0, \ldots, N$. If $X = 0$, no packet loss occurs. If $X \geq 1$, then $X - 1$ packets are lost, since only one packet is served. Hence, if Y is the number of lost packet directed to U, with $Y = \max(0, X - 1)$, then the average number of lost packets are:

$$E[Y] = \sum_{x=2}^{N} (x - 1) \binom{N}{x} \left(\frac{p}{N}\right)^x \left(1 - \frac{p}{N}\right)^{N-x}$$

obtained by considering only when the number of lost packets is between 1 and $N - 1$. $E[Y]$ can be computed by recalling that for the binomial distribution, with parameters (q, N) and $0 \leq q \leq 1$, it holds:

$$\sum_{x=0}^{N} \binom{N}{x} q^x (1 - q)^{N-x} = 1 \qquad \sum_{x=0}^{N} x \binom{N}{x} q^x (1 - q)^{N-x} = Nq$$

N	p=0.1	p=0.5	p=1.0
8	0.034	0.77	2.74
16	0.073	1.63	5.70
256	1.24	27.2	94.0

Table 3.1: Average number of lost cells $NE[Y]$ (ex. 82)

N	Max throughput	Max port speed
8	0.656	656 Mbit/s
16	0.644	644 Mbit/s
256	0.633	633 Mbit/s
∞	0.632	632 Mbit/s

Table 3.2: Maximum throughput for 1Gbps ports switch (ex. 82)

Now:

$$E[Y] = N\frac{p}{N} - \binom{N}{1}\left(\frac{p}{N}\right)\left(1-\frac{p}{N}\right)^{N-1} - 1 + \left(\left(1-\frac{p}{N}\right)^{N} + N\left(\frac{p}{N}\right)\left(1-\frac{p}{N}\right)^{N-1}\right) = \left(1-\frac{p}{N}\right)^{N} - (1-p)$$

The average total number of cells lost in the switch is given by:

$$NE[Y] = N\left(\left(1-\frac{p}{N}\right)^{N} - (1-p)\right)$$

which is shown in Table 3.1 and, for $N \to \infty$, goes to $N[e^{-p} - (1-p)]$.

The single port throughput is equal to the probability that an output is served $P(X \geq 1)$:

$$P(X \geq 1) = 1 - P(X = 0) = 1 - \left(1-\frac{p}{N}\right)^{N}$$

which is shown in Table 3.2. The limiting throughput for each single port is:

$$\lim_{N\to\infty} 1 - \left(1-\frac{p}{N}\right)^{N} = 1 - e^{-p} = 1 - e^{-1} = 63\%$$

where p has been set equal to 1 to compute the limiting throughput.

★★★

Exercise 83 (out of scope for the program 2021/22)
Consider a bufferless switch. Show analytically that the maximum achievable throughput is around 63%, specifying in details all the assumptions to get such a result.

Why the throughput is larger than the throughput achievable in an input queued switch with a single queue per input?

Solution: See solution of Ex. 82.

★★★

Exercise 84 (out of scope for the program 2021/22)
Consider a slotted bufferless switch of size $N \times M$, with any N and M, comprising all the possible three cases: $N = M$, $N > M$ and $N < M$. When an output contention occurs among different packets, one packet at random is transferred across the switch. Assume that the arrival process is Bernoulli i.i.d. being $\rho \in [0, 1]$ the normalized average load at an input. The traffic is uniformly distributed across all the inputs and outputs.

1. *Compute the admissibility conditions for ρ*
2. *Compute the throughput achievable in function of ρ, describing in details all the steps in the derivation.*
3. *Assume $\alpha = N/M$ fixed and $N \to \infty$. What is the maximum throughput achievable? If needed, recall that*

$$\lim_{x\to\infty}\left(1+\frac{a}{x}\right)^x = e^a$$

Solution:

1. For $M < N$, it should be $\rho < M/N$, and for $M \geq N$ it should be $\rho \leq 1$.
2. Observe that ρ/M is the probability that an output receives a packet during a generic timeslot. Let X be the number of packets arrived for a specific output.

$$P(X=0) = \left(1-\frac{\rho}{M}\right)^N$$

   The throughput T can be computed as

$$T = P(X \geq 1) = 1 - P(X=0) = 1 - \left(1-\frac{\rho}{M}\right)^N$$

3. If $N = \alpha M$

$$T = 1 - \left(1-\frac{\rho}{M}\right)^{\alpha M} \to 1 - e^{-\rho\alpha}$$

   If $N < M$ (i.e. $\alpha < 1$), the maximum admissible load is $\rho = 1$ and $T \to 1 - e^{-\alpha}$. If $N \geq M$ (i.e. $\alpha \geq 1$), the maximum admissible load is $\rho = 1/\alpha$ and $T \to 1 - e^{-1} \approx 0.63$.

★★★

Exercise 85 (out of scope for the program 2021/22)
Consider a 4×4, bufferless *switch, fed by non-uniform Bernoulli i.i.d. arrivals according to the following rate matrix:*

$$\Lambda = \rho \begin{pmatrix} 1/3 & 1/3 & 1/6 & 1/6 \\ 1/3 & 1/3 & 1/6 & 1/6 \\ 1/3 & 1/3 & 1/6 & 1/6 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where ρ is the normalized load at each input. Assume that contentions among packets directed to the same output are solved at random. Compute analytically

1. *the traffic admissibility conditions;*
2. *the throughput measured for each output port in function of ρ;*
3. *the maximum throughput achievable under admissible traffic.*

Solution:
The admissibility conditions are $\rho \leq 1$.
Consider a generic timeslot. Let X_1 be the number of cells arrived and directed to output 1 and 2 and let X_2 be the number of cells arrived and directed to output 3 and 4.

$$P(X_1=0) = \left(1-\frac{\rho}{3}\right)^3 \qquad P(X_2=0) = \left(1-\frac{\rho}{6}\right)^3$$

Now the throughput for outputs 1 and 2 is:

$$T_1(\rho) = P(X_1 \geq 1) = 1 - P(X_1 = 0) = 1 - \left(1 - \frac{\rho}{3}\right)^3$$

and for outputs 3 and 4:

$$T_2(\rho) = P(X_2 \geq 1) = 1 - P(X_2 = 0) = 1 - \left(1 - \frac{\rho}{6}\right)^3$$

The maximum throughput is achieved for $\rho = 1$:

$$T_1(1) = 1 - (1 - 1/3)^3 \approx 0.70 \qquad T_2(1) = 1 - (1 - 1/6)^3 \approx 0.42$$

★★★

Exercise 86 (out of scope for the program 2021/22)
Consider a bufferless switch of size $N \times M$, running on a slotted fashion. At each timeslot, a broadcast packet (i.e., directed to all the M outputs) arrives at each input with probability p.

1. *compute the throughput in function of p*
2. *compute the admissibility condition for p*
3. *compute the maximum throughput under admissible traffic*

Solution:

1. $(1 - (1 - p)^N)$
2. $p \leq 1/N$
3. $1 - e^{-1} \approx 0.63$

★★★

Exercise 87 (out of scope for the program 2021/22)
Consider a generic bufferless $N \times M$ *switch with synchronous operation: at most one cell is transferred for each timeslot from each input and to each output. The traffic is multicast and uniformly distributed among all inputs and the fanout set is uniformly distributed among all the outputs. Fanout splitting is allowed, i.e. a cell can be transferred in one timeslot to a subset of its destinations.*

Let ρ be the arrival probability of a cell at each input during one timeslot. Given a multicast cell, let q be the probability that the cell is directed to a specific output.

1. *What is the distribution of the fanout (i.e. the number of destinations) of a generic cell?*
2. *What is the average fanout f of a cell?*
3. *What is the average offered load at each output?*
4. *Under which conditions the traffic is admissible?*

Now fix the attention to a specific output.

1. *What is the distribution of the number of cells directed to that specific output?*
2. *What is the average throughput as a function of ρ and f?*

3. *What is the maximum throughput under admissible traffic for finite N and M? What about taking the limits for N and M going to infinity?*

Solution: Let X be the fanout of a generic cell

$$P(X = x) = \binom{M}{x} q^x (1-q)^{M-x} \qquad 0 \le x \le M$$

Note that the model implies that, with probability $(1-q)^M$, the packet fanout is null. In theory, $P(X = x)$ could be modified to avoid this case, but we have preferred to keep this case for the sake of simpler theory. Now, the average fanout is $f = E[X] = qM$ and the traffic is admissible if

$$N\rho f/M < 1 \qquad \Rightarrow \qquad N\rho q < 1$$

Let Y be the number of cells directed for a specific output. Since ρq is the probability that an input has a cell destined to a specific output:

$$P(Y = y) = \binom{N}{y} (\rho q)^y (1-\rho q)^{N-y} \qquad 0 \le y \le N$$

The probability that no cell is received for an output is:

$$P(Y = 0) = (1-\rho q)^N$$

The average throughput, seen at any output, is equal to the probability that an output is busy

$$T = P(Y > 0) = 1 - P(Y = 0) = 1 - (1-\rho q)^N = 1 - \left(1 - \rho \frac{f}{M}\right)^N$$

Note that the maximum throughput is achieved for $\rho = 1/(qN)$:

$$T_{max} = 1 - \left(1 - \frac{1}{N}\right)^N$$

which goes to $1 - e^{-1} \approx 63\%$, independently of M, for $N \to \infty$.

3.1.2 Input queued switch with single FIFO

Exercise 88 (out of scope for the program 2021/22)
Consider a 2×2 input queued switch, with one queue for each input, fed by uniform Bernoulli i.i.d. traffic.

1. *Find analytically the maximum achievable throughput. Define all the adopted notation.*
2. *Discuss qualitatively the delay performance in function of the average load.*

Is it possible to devise another admissible traffic scenario for which the throughput is lower than in the above case?

★★★

Exercise 89 (out of scope for the program 2021/22)
Consider an $N \times N$ input queued switch with a single FIFO queue per input port. Find an admissible traffic pattern, with load $\rho = 1$, for which the maximum throughput is order of $1/N$ and prove such result.

Solution: Let $A_i(n)$ be the destination of the packet arrived at input i, with $i \in [1, \ldots, N]$, at timeslot n. Fix k to a generic value. During the observation interval of Nk timeslots, the traffic pattern is the following, for each input i:

- $A_i(1) = \ldots = A_i(k) = 1$
- $A_i(k+1) = \cdots = A_i(2k) = 2$
- $A_i(2k+1) = \cdots = A_i(3k) = 3$
- $\ldots$
- $A_i((N-1)k+1), \ldots, A_i(Nk) = N$.

Let $x(j)$ be the time when the last packet, destined for output j, is served. We assume a round robin service at each output, in the case of contention for the same output; hence,

- $x(1) = kN$
- $x(2) = x(1) + (kN - (N-1))$
- $x(j) = x(j-1) + (kN - (N-1))$

from which:

$$x(N) = x(1) + (kN - (N-1))(N-1) = kN^2 - (N-1)^2$$

The timeslot when the last packet arrives for output N is kN. Assume that such packet is arrived at input N. Then, the maximum throughput T achieved by input N is:

$$T = \frac{kN \ \text{ served packets}}{x(N) \ \text{ timeslots}} = \frac{kN}{kN^2 - (N-1)^2}$$

from which, if we want a throughput close to $1/N$ (thus, approaching 0 as N increases), we can set:

$$T \leq \frac{1+\epsilon}{N}$$

and

$$k \geq \frac{1+\epsilon}{\epsilon} \left(\frac{N-1}{N} \right)^2$$

★★★

Exercise 90 (out of scope for the program 2021/22)
Consider a 2×2 input queued switch with just one single FIFO queue per each input fed by Bernoulli i.i.d. arrivals with the following arrival matrix:

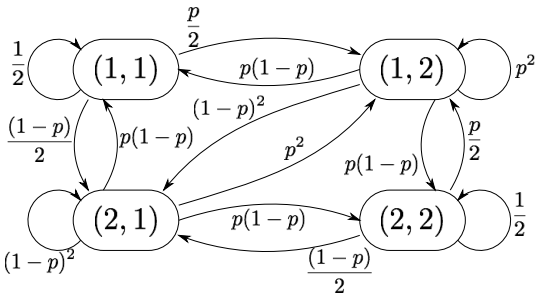
$$\Lambda = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix}$$

1. *Show the Markov chain to compute the maximum throughput, after having properly defined each state.*
2. *Write the formula to compute the maximum throughput in function of the stationary probability of each state.*
3. *What is the throughput for $p \to 1$?*
4. *What is the throughput for $p \to 0$?*

5. *What is the throughput for $p = 0.5$?*

Solution:

The state of the Markov chain is (O_1, O_2) where O_i is the destination of the packet at the head of the queue at input i. The Markov chain is the following:



The throughput can be evaluated as:

$$T = \frac{1\Pi_{(1,1)} + 1\Pi_{(2,2)} + 2\Pi_{(1,2)} + 2\Pi_{(2,1)}}{2}$$

where two states have been aggregated into a single one.

Without solving the Markov chain, but just understanding the traffic fed to the switch, when either $p = 0$ or $p = 1$ the throughput is always 100% ($T = 1$) since no conflict is present.

In the case of $p = 0.5$, the traffic is uniform and the throughput is $T = 0.75$.

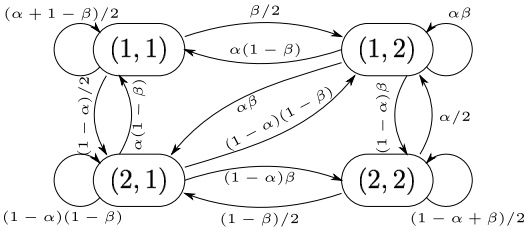
★★★

Exercise 91 (out of scope for the program 2021/22)
Consider a 2×2 input queued switch with just one single FIFO queue per each input fed by Bernoulli i.i.d. arrivals with the following arrival matrix:

$$\Lambda = \begin{bmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{bmatrix}$$

1. *Show the Markov chain to compute the maximum throughput, after having properly defined each state.*
2. *Write the formula to compute the maximum throughput in function of the stationary probability of each state.*
3. *What is the throughput for $\alpha \to 1$ and $\beta \to 1$?*
4. *What is the throughput for $\alpha \to 0.5$ and $\beta \to 0.5$?*
5. *What is the throughput for $\alpha \to 0$ and $\beta \to 0$?*

Solution: The state of the Markov chain is (O_1, O_2) where O_i is the destination of the packet at the head of the queue at input i. The Markov chain is the following:



The the remaining of the exercise is as Ex. 90

★★★

Exercise 92 (out of scope for the program 2021/22)
Show how to prove the 58% throughput result for an input queued switch.

1. *describe all the assumptions on the input traffic, the switch queueing structure, and the scheduling algorithm*
2. *describe the "derived" queueing model considered in the proof to obtain the final results*
3. *obtain the final result.*

Recall that the average queue size $E[X]$ of a discrete M/D/1 queue is:

$$E[X] = \frac{E[A^2] + E[A] - 2(E[A])^2}{2(1 - E[A])}$$

where A is a discrete random variable equal to the number of packet arrivals during the current timeslot.

★★★

Exercise 93 (out of scope for the program 2021/22)
Consider a $N \times N$ input queued switch with just one single FIFO queue per each input, fed by admissible Bernoulli i.i.d. arrivals.

1. *Explain in details the meaning of admissible Bernoulli i.i.d. arrivals.*
2. *Describe the performance in terms of average delay and throughput, under uniform traffic. Draw the corresponding two performance curves: throughput vs load, delay vs load.*
3. *Describe a non-uniform traffic scenario in which the throughput is always maximum. Draw the two performance curves, as above.*
4. *Describe a non-uniform traffic scenario (also non-Bernoulli) in which the throughput is very small. Draw the two performance curves, as above.*

3.1.3 Generic switches with input and/or output queueing

Exercise 94 (out of scope for the program 2021/22)
Consider a packet switch.

1. *Define the work-conservation property.*
2. *Is an output queued switch work-conserving? Prove it formally.*
3. *Is an input queued switch work-conserving? Prove it formally.*
4. *Which performance index is mainly affected by the work-conservation property?*

Solution:

1. A time-slotted packet switch is work-conserving if, for each timeslot, whenever at least one packet arrives destined to an output, this output will be busy transmitting a packet to the output interface. Thus, if a packet arrives and it is not transmitted at the output interface, then this output is busy transmitting another packet.

2. An output queued switch is work-conserving by construction, since the speedup is sufficient to transfer all the packets present at the inputs to the outputs. Hence, it cannot happen that an output interface will be idling if at least one packet destined for it is available.

3. An input queued switch is not work-conserving. Consider, for example, the following sequence of packets arriving to a 2×2 switch, using the notation (input, output), with the outputs denoted as A e B.

   - $t = 0$: (1,A), (2,A), and only (2,A) is transmitted to output A;
   - $t = 1$: (1,B); only one between (1,A) and (1,B) is transmitted to the output; thus, one output interface will be busy, whereas the other will be idling even if a packet is available and destined to it.

4. The work-conservation property affects mainly the average delay.

★★★

Exercise 95 (out of scope for the program 2021/22)
Consider a combined input output queued (CIOQ) switch, with speedup 2.

- *Does it achieve 100% throughput? How? At which complexity?*
- *Does it achieve work conservation? How? At which complexity?*
- *Does it achieve perfect OQ emulation? How? At which complexity?*

Solution: To achieve 100% throughput, any maximal size matching is sufficient; the complexity is around $O(N^2)$. At the expenses of a larger complexity, LOOFA achieves work conservation. With much higher complexity and an algorithm based on stable marriage problem, it is also possible to achieve OQ emulation.

★★★

Exercise 96 (out of scope for the program 2021/22)
Consider a Combined Input Output Queued (CIOQ) switch.

1. *Define the work-conservation property*
2. *Define the output-queued (OQ) emulation property*
3. *Describe the algorithm to obtain work-conservation for speedup 2*
4. *Describe the algorithm to obtain OQ emulation for speedup 4*

★★★

Exercise 97 (out of scope for the program 2021/22)
Consider an 2×2 switch with a single queue for input, with speedup S. The offered load is Bernoulli i.i.d., uniformly distributed among all the inputs and the outputs.

1. *Compute the maximum achievable throughput for $S = 1$.*
2. *Compute the maximum achievable throughput for $S = 2$.*
3. *Draw, qualitatively, on the same graph the curves of the average delay in function of offered load in the cases: $S = 1$ e $S = 2$.*

Solution:

1. $S = 1$: 75% (see the class notes);
2. $S = 2$: 100% (equivalent to an OQ);
3. See the class notes.

★★★

Exercise 98 (out of scope for the program 2021/22)
Consider an input queued switch $N \times M$ fed by multicast traffic.

1. *Define the optimal queueing structure. How many queues are needed?*
2. *Describe a counterexample showing that the input queued switch achieves the maximum throughput lower that an output queued switch, in terms of maximum throughput, regardless the adopted queueing structure.*

★★★

Exercise 99 (out of scope for the program 2021/22)
Show how to compute the average delay for an output queued architecture, as function of λ, the average single-input load ($0 \leq \lambda \leq 1$), under uniform i.i.d. Bernoulli traffic. Recall that the average queue size $E[X]$ of a discrete M/D/1 queue is:

$$E[X] = \frac{E[A^2] + E[A] - 2(E[A])^2}{2(1 - E[A])}$$

where A is a discrete random variable equal to the number of packet arrivals during the current timeslot. Finally, draw the graph of the average delay as function of λ.

3.2 Packet Scheduling in Input Queued Switches

3.2.1 Scheduling algorithms for unicast traffic

Exercise 100 (out of scope for the program 2021/22)
Consider an input queued switch with Virtual Output Queueing fed by admissible Bernoulli i.i.d. (ABIID) traffic. The adopted scheduler computes the maximum size matching at each timeslot.

1. *Explain in details the meaning of "admissible" traffic*
2. *Explain in details the meaning "Bernoulli i.i.d." traffic*
3. *Describe an ABIID traffic pattern under which the scheduler achieves 100% throughput*
4. *Describe an ABIID traffic pattern under which the scheduler does not achieve 100% throughput and prove such property, by explaining all the required assumptions.*

★★★

Exercise 101 (out of scope for the program 2021/22)
Consider a 4×4 input queued switch with VOQ. Report four examples, one for each below listed case, of the queue occupancies such that:

1. *the maximum size matching is different from the maximal size matching*

2. *the maximum weight matching is different from the maximal weight matching*

3. *the maximum size matching is different from the maximum weight matching*

4. *the maximal size matching is different from the maximal weight matching*

Solution: Let R be the request matrix, corresponding to the VOQ occupancy. Let MWM/mWM be the maximum/maximal weight matching, and MSM/mSM be the maximum/maximal size matching. Given a matching π, let $\pi(i)$ be the output connected to input i; if i is not connected, $\pi(i) = -$.

1. $R = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$: $\pi_{MSM} = (1, 2, 3, 4)$, $\pi_{mSM} = (2, -, 3, 4)$.

2. $R = \begin{bmatrix} 2 & 3 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$: $\pi_{MWM} = (1, 2, 3, 4)$, $\pi_{mWM} = (2, -, 3, 4)$.

3. $R = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$: $\pi_{MSM} = (1, 2, 3, 4)$, $\pi_{MWM} = (2, -, 3, 4)$.

4. $R = \begin{bmatrix} 2 & 3 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$: $\pi_{mSM} = (1, 2, 3, 4)$, $\pi_{mWM} = (2, -, 3, 4)$.

★★★

Exercise 102
Describe in pseudo-code a scheduling algorithm for input queued switches, that is maximal and approximates the maximum size matching*.*

Does it obtain always 100% throughput? Why? (question out of scope for the program 2021/22)

Solution:

The pseudocode is available in Fig. 3.1.

The algorithm does not obtain 100% throughput since it approximates an algorithm which does not achieve it neither (see the counterexample in the class notes). Further, a maximal algorithm is able to obtain 100% throughput only when the available speedup is 2.

★★★

Exercise 103
Describe in pseudo-code a scheduling algorithm for input queued switches, that is maximal and approximates the maximum weight matching*.*

Does it obtain always 100% throughput? Why? Under which traffic scenarios it obtains the maximum throughput? (questions out of scope for the program 2021/22)

Solution: The pseudocode is available in Fig. 3.2.

The algorithm does not obtain 100% throughput since a maximal algorithm is able to achieve 100% throughput when the speedup is at least 2. It obtains 100% throughput under uniform Bernoulli i.i.d. traffic.

```
void scheduler(int **X) { // X is matrix of size NUM_PORTS*NUM_PORTS
    // X[in][out]=queue lenght for VOQ[in][out]
    int in,out;
    int matching[NUM_PORTS]; // matching[in]=out, otherwise -1
    unsigned char out_reserved[NUM_PORTS]; // =TRUE, FALSE

    // init matching and reservation vectors
    for (in=0; in<NUM_PORTS; in++) {
        matching[in]=-1;
        out_reserved[in]=FALSE;
    }
    for (in=0; in<NUM_PORTS; in++) {
        for (out=0; out<NUM_PORTS; out++) {
            if (X[in][out]>0 && out_reserved[out]==FALSE) {
                matching[in]=out;
                out_reserved[out]=TRUE;
                break;
            }
        }
    }
    // now matching contains the desired matching
}
```

Figure 3.1: Pseudocode for Ex. 102

```
void scheduler(int **X) { // X is matrix of size NUM_PORTS*NUM_PORTS
    // X[in][out]=queue length for VOQ[in][out]
    int in,out,out_where_max_len,max_len;
    int matching[NUM_PORTS]; // matching[in]=out, otherwise -1
    unsigned char out_reserved[NUM_PORTS]; // =TRUE, FALSE

    // init matching and reservation vectors
    for (in=0; in<NUM_PORTS; in++) {
        matching[in]=-1;
        out_reserved[in]=FALSE;
    }
    for (in=0; in<NUM_PORTS; in++) {
        // find the maximum queue among unreserved outputs
        max_len=0;
        for (out=0; out<NUM_PORTS; out++) {
            if (X[in][out]>max_len && out_reserved[out]==FALSE) {
                max_len=X[in][out];
                out_where_max_len=out;
            }
        }
        // store the maximum queue
        if (max_len>0) {
            matching[in]=out_where_max_len;
            out_reserved[out_where_max_len]=TRUE;
        }
    }
    // now matching contains the desired matching
}
```

Figure 3.2: Pseudocode for Ex. 103

★★★

Exercise 104 (out of scope for the program 2021/22)
Consider an input queued switch, with Virtual Output Queueing and without speedup. The scheduling algorithm must maximize, at any timeslot, the number of packets to transfer from the inputs to the outputs.

1. *Show an example of scheduling algorithm which can be used.*
2. *Is the switch work-conserving? Why? Prove it, if possible.*

3. *Does the switch obtain always 100% throughput? Why? Prove it, if possible.*

★★★

Exercise 105
Consider an input queued switch, with Virtual Output Queueing and without speedup.

1. *Describe (better if in pseudo-code) any maximal scheduling algorithm that approximates the maximum size matching.*
2. *How does it behave, in terms of throughput and delay, with respect to an output queued switch? (question out of scope for the program 2021/22)*

★★★

Exercise 106
Consider a $N \times M$ input queued switch with Virtual Output Queueing and QoS support. For a generic (input-i,output-j) pair there exist two queues: VOQ_{ij}^H for high priority traffic (e.g. VoIP) and VOQ_{ij}^L for low priority traffic (e.g. web). An input traffic classifier sends the incoming packets to the correct queue.

1. *Write in pseudocode a greedy algorithm to schedule the transmissions across the switching fabric, to maximize the number of high priority packets that are selected at each timeslot. Let $H[i][j]$ be the occupancy of VOQ_{ij}^H and let $L[i][j]$ be the occupancy of VOQ_{ij}^L.*
2. *Discuss the scheduler performance in terms of throughput. (question out of scope for the program 2021/22)*
3. *Is it possible that low priority traffic will be starved indefinitely by high priority traffic? Motivate your answer with an example.*

★★★

Exercise 107
Consider a $N \times M$ input queued switch with Virtual Output Queueing and supporting Strict Priority Queueing based on C classes. For each (input-i,output-j) pair there exist C queues: $[VOQ_{ij}^c]_{c=1}^C$, with decreasing priority level, where $c = 1$ refers to the highest priority traffic and $c = C$ refers to the lowest priority traffic (i.e., best effort traffic). Therefore, $C \times N \times M$ queues are present in total. We assume that an input traffic classifier sends the incoming packets to the correct queue.

1. *Write in pseudocode a greedy algorithm to schedule the transmissions across the switching fabric, to be maximal and to maximize the number of higher priority packets that are selected at each timeslot. Define all the data structures adopted in your code.*
2. *Does the scheduler obtain always the maximum throughput under admissible Bernoulli i.i.d. traffic? Motivate your answer. (question out of scope for the program 2021/22)*
3. *Is it possible that low priority traffic will be starved indefinitely by high priority traffic? Motivate your answer with an example.*

★★★

Exercise 108
Consider a $N \times M$ input queued switch with Virtual Output Queueing, running in a network where each packet is tagged with the entrance time in the network (i.e. the packet is tagged at the first switch traversed in the network with a timestamp corresponding to the current time). Each VOQ is not FIFO, but stores the packets in increasing order of timestamp, such that the head-of-line packet at each VOQ is the oldest in the network. Let VOQ_{ij} be the queue corresponding to input i and output j and let $H[i][j]$ be its occupancy. Let T_{ij} be the timestamp of the head-of-line packet corresponding to VOQ_{ij}, represented by an integer number corresponding to the absolute time when the packet entered the network.

1. *Write in pseudocode a greedy algorithm to schedule the transmissions across the switching fabric, to maximize the number of oldest packets that are selected at each timeslot.*
2. *Discuss the scheduler performance in terms of throughput. (question out of scope for the program 2021/22)*
3. *Is it possible that a packet will be starved indefinitely in the switch? Motive your answer.*

★★★

Exercise 109
Consider a $N \times M$ input queued switch with Virtual Output Queueing for data centers. To minimize the flow completion time, the scheduler transfers at highest priority the packets corresponding to shortest flows. Let $F[i][j]$ be the flow length of the head-of-the-line packet of the VOQ from input i to output j.

1. *Write in pseudocode a greedy algorithm to schedule the transmissions.*
2. *Define all the required data structures, providing a detailed description of their use and meaning.*
3. *Can the packets of a specific flow starve forever? Motivate in details your answer.*

★★★

Exercise 110
Describe in pseudo-code a scheduling algorithm for an $A \times B$ input queued switch, designed to reduce the flow completion time in data centers. Such algorithm is maximal and gives higher priority to the head-of-the-line (HoL) packets with less residual packets in the corresponding flow. Let $P[i][j]$ be the residual number of packets for the flow of the HoL packet at VOQ at input i and destined to output j. Define all the required data structures.

Could be there any starvation problem for the flows? Why?

★★★

Exercise 111
Consider a Top-of-Rack switch with 42 ports, in which 4 ports (denoted as "gold" ports) are adopted to connect to the spine switches and the remaining ports are devoted to the interconnection of the servers within the rack. A QoS policy assures that all the traffic that is directed to any gold port **or** *is received by any gold port must be transferred at the highest priority with respect to the remaining traffic.*

Assume now that the traffic is only unicast and the switch is implemented through an input queued architecture with Virtual Output Queueing. The scheduler is maximal and implements a strict priority policy on the gold ports.

- *Define and describe all the data structures required to define and solve the scheduling problem.*
- *Describe in pseudocode the scheduling algorithm.*
- *Discuss the expected performance in terms of throughput and delays, under a generic traffic pattern. (question out of scope for the program 2021/22)*

Solution:

```
void scheduler(int **X, int NUM_PORTS, int NUM_GOLD_PORTS) {
    // X is matrix of size NUM_PORTS*NUM_PORTS; X[in][out]=queue length for VOQ[in][out]
    // gold ports are NUM_GOLD_PORTS and correspond to the first NUM_GOLD_PORTS ports

    int in,out;
    int matching[NUM_PORTS]; // matching[in]=out, otherwise -1
    bool out_reserved[NUM_PORTS],in_reserved[NUM_PORTS];  // boolean reservation vectors

    // init matching and reservation vectors
    for (in=0; in<NUM_PORTS; in++) {
        matching[in]=-1;
        out_reserved[in]=in_reserved[in]=false;
    }
    // Step 1: match gold input ports to any output port
    for (in=0; in<NUM_GOLD_PORTS; in++) {
        for (out=0; out<NUM_PORTS; out++) {
            if (out_reserved[out]) { // skip already reserved outputs
                continue;
            }
            if (X[in][out]>0) {
                // if the VOQ is non empty, store the matching and reserve the inputs/outputs
                matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                break;
            }
        }
    }
    // Step 2: match gold output ports to any input port
    for (out=0; out<NUM_GOLD_PORTS; out++) {
        if (out_reserved[out]) { // skip already reserved outputs
                continue;
        }
        for (in=0; in<IN_PORTS; in++) {
            if (in_reserved[in]) { // skip already reserved inputs
                continue;
            }
            if (X[in][out]>0) {
                // if the VOQ is non empty, store the matching and reserve the inputs/outputs
                matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                break;
            }
        }
    }
    // Step 3: match remaining ports
    for (in=NUM_GOLD_PORTS; in<NUM_PORTS; in++) {
        if (in_reserved[in]) { // skip already reserved inputs
                continue;
        }
        for (out=NUM_GOLD_PORTS; out<NUM_PORTS; out++) {
            if (out_reserved[out]) { // skip already reserved outputs
                continue;
            }
            if (X[in][out]>0) {
                // if the VOQ is non empty, store the matching and reserve the inputs/outputs
                matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                break;
            }
        }
    }
    // now matching contains the desired matching
}
```

Another possibility for the pseudocode:

```
void scheduler(int **X, int NUM_PORTS, int NUM_GOLD_PORTS) {
    // X is matrix of size NUM_PORTS*NUM_PORTS; X[in][out]=queue length for VOQ[in][out]
    // gold ports are NUM_GOLD_PORTS and correspond to the first NUM_GOLD_PORTS ports

    int in,out;
    int matching[NUM_PORTS]; // matching[in]=out, otherwise -1
    bool out_reserved[NUM_PORTS],in_reserved[NUM_PORTS];  // boolean reservation vectors

    // init matching and reservation vectors
    for (in=0; in<NUM_PORTS; in++) {
        matching[in]=-1;
        out_reserved[in]=in_reserved[in]=false;
    }
    // Step 1: match gold input or ports
    for (in=0; in<NUM_PORTS; in++) {
        for (out=0; out<NUM_PORTS; out++) {
            // check if the input or the output port is a gold port
            if (in<NUM_GOLD_PORTS OR out<NUM_GOLD_PORTS) {
                // check if the VOQ is not empty the the output is available
                 if (X[in][out]>0 AND !out_reserved[out]) {
                      // store the matching and reserve both the inputs and outputs
                      matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                      break; // consider a new input
                }
        }
    }
    // Step 2: match all the remaining ports
    for (in=0; in<NUM_PORTS; in++) {
        if (in_reserved[in]) { // skip already reserved input
                continue;
        }
        for (out=0; out<NUM_PORTS; out++) {
                // check if the VOQ is not empty the output is available
                 if (X[in][out]>0 AND !out_reserved[out]) {
                        // store the matching and reserve both the inputs and outputs
                        matching[in]=out; out_reserved[out]=in_reserved[in]=true;
                        break; // consider a new input
                }
        }
    }
    // now matching contains the desired matching
}
```

★★★

Exercise 112 (out of scope for the program 2021/22)
Consider two scheduling algorithms S_1 and S_2, for an $N \times N$ input queued switch, achieving 100% throughput. Let S_1 compute a maximal weight matching. Let S_2 exploit memory from past matchings.

1. *Describe in pseudo-code the algorithms S_1 and S_2, assuming that it is already available a function, returning a random matching, declared as follows:*

   ```
   int *create_random_matching(void)
   ```

2. *Describe the sufficient conditions for S_1 and S_2 to obtain 100% throughput.*

3. *Compute the approximated computational complexity for S_1 and S_2 in terms of elementary operations, knowing that the minimum complexity to find a random matching is $O(N \log N)$.*

★★★

Exercise 113 (out of scope for the program 2021/22)
Show that the maximum weight matching (MWM) obtains 100% throughput in an input queued switch, under admissible uniform Bernoulli i.i.d. traffic. Let $X = [x_{ij}]$ be the matrix with x_{ij} the length of the virtual output queue from input i to output j. Let $D = [d_{ij}]$ be the MWM computed on X. Let $\Lambda = [\lambda_{ij}]$ matrix with the average arrival rates. Show the key step in the proof, i.e. it exists $\epsilon > 0$ such that:

$$\lim_{B \to +\infty} \frac{\sum_{ij} \lambda_{ij} x_{ij} - \sum_{ij} d_{ij} x_{ij}}{B} < -\epsilon$$

where $B = \sum_{ij} x_{ij}$. Hints:

- *use the Birkhoff-von Neumann theorem on Λ;*
- *exploit the relation between $\sum_{ij} x_{ij}$ with the weight of the MWM computed on X.*

Solution: See the class notes.

★★★

Exercise 114 (out of scope for the program 2021/22)
The maximum weight matching (MWM) is claimed to be an optimal scheduling algorithm.

1. *for which switching architecture?*
2. *in which sense?*
3. *under which conditions?*
4. *Compare its performance with an output queued (OQ) architecture regarding throughput and delay, motivating the answer.*
5. *Is it commonly implemented in routers? Why?*

Solution: The MWM algorithm is optimal in the sense that it achieves 100% throughput under any admissible i.i.d. Bernoulli arrival traffic (more generally, if the traffic follows the law of large number, the result still holds in a weaker sense). The switching architecture considered is the input queued (IQ), with a VOQ queueing system.

MWM achieves the same throughput of an OQ, since also an OQ switch achieves the maximum throughput by construction. MWM shows higher average delays than an OQ, since the IQ is non-work-conserving, whereas the OQ is work-conserving by construction. By explaining the counterexample seen during the class, it is possible to clarify this concept and show that the delay is actually larger for the IQ switch adopting MWM.

MWM has not been implemented in routers mainly because: (i) it is too complex to implement (requires $O(N^3)$ iterations, it cannot be parallelized and pipelined efficiently), (ii) it is based on a queue metrics (the queue length) which might react with the congestion control of TCP flows and create problems of starvation for some TCP flows.

★★★

Exercise 115 (out of scope for the program 2021/22)
Prove that the weight of a greedy maximum weight matching (GWM) is at least equal to half the weight of the maximum weight matching (MWM). In other words, $W(GWM) \geq \frac{1}{2} W(MWM)$.

In the proof, denote by E the set of edges in the bipartite graph, by G the sub-set of edges selected by the GWM, and by M the sub-set of edges selected by the MWM scheduler.

★★★

Exercise 116 (out of scope for the program 2021/22)
Prove that the weight of a greedy maximal weight matching (GWM) is at least equal to half the weight of the maximum weight matching (MWM). In other words,

$$W(GWM) \geq \frac{1}{2} W(MWM) \tag{3.1}$$

In the proof, denote by E the set of edges in the bipartite graph, by G the sub-set of edges selected by the GWM, and by M the sub-set of edges selected by the MWM scheduler.

What is the main consequence of (3.1) in terms of maximum throughput achievable by GWM?

★★★

Exercise 117 (out of scope for the program 2021/22)
Consider the proof showing that a greedy maximal weight matching (GMWM) obtains 100% throughput in an input queued switch with speedup $s \geq 2$, under admissible uniform Bernoulli i.i.d. traffic. Let $X = [x_{ij}]$ be the matrix with x_{ij} the length of the virtual output queue (VOQ) from input i to output j. Let $D = [d_{ij}]$ be the GMWM computed on X. Let $\Lambda = [\lambda_{ij}]$ matrix with the average arrival rates. Show the key step in the proof, i.e. it exists $\epsilon > 0$ such that:

$$\lim_{B \to +\infty} \frac{\sum_{ij} \lambda_{ij} x_{ij} - \sum_{ij} s\, d_{ij} x_{ij}}{B} < -\epsilon$$

where $B = \sum_{ij} x_{ij}$ and s is the speedup. Hints:

- *use the Birkhoff-von Neumann theorem on Λ;*
- *exploit the relation between the weight of GMWM and the weight of MWM (Maximum Weight Matching);*
- *exploit the relation between $\sum_{ij} x_{ij}$ and the weight of MWM computed on X.*

3.2.2 Scheduling algorithms for variable size packets

Exercise 118 (out of scope for the program 2021/22)
Consider an input queue switch fed by variable size packets.

1. *Describe the overall architecture highlighting all the involved queuing systems*
2. *Describe the cell-mode and packet-mode schemes for scheduling the packets*
3. *Describe a traffic pattern in which packet-mode outperforms cell-mode in terms of average delay*
4. *Describe a traffic pattern in which cell-mode outperforms packet-mode in terms of average delay*
5. *Compare the two schemes in terms of throughput*

★★★

Exercise 119 (out of scope for the program 2021/22)
Describe the packet-mode *scheduling approach in input queued switches and discuss its performance in terms of throughput and delays, motivating the answer.*

★★★

Exercise 120 (out of scope for the program 2021/22)
Consider an input queued switch fed by variable size packets.

1. *Describe the overall architecture highlighting all the involved queuing systems*
2. *Describe both in words and in pseudo-code a "cell mode" scheduler*
3. *Describe both in words and in pseudo-code a "packet mode" scheduler*
4. *Compare the two schedulers in terms of throughput and delays*

★★★

Exercise 121 (out of scope for the program 2021/22)
Consider an input queued switch fed by variable size packets.

1. *Describe the overall architecture highlighting all the involved queuing systems*
2. *Describe in pseudocode a "packet mode" scheduler*
3. *Discuss its performance in terms of throughput and delays*

★★★

Exercise 122 (out of scope for the program 2021/22)
Consider an IP router, based on a switching matrix transferring data units (called "cells") of 64 bytes.

1. *Describe the internal switching architecture of the router, able to transfer IP packets of variable size, between 40 and 1518 bytes.*
2. *If the switching fabric is input queues and no speedup is allowed, which scheduling algorithms can be used? What are the complexity and the performance?*

★★★

Exercise 123 (out of scope for the program 2021/22)
Consider a slotted input queued switch, of size $N \times M$ and with Virtual Output Queueing. The switch is fed by variable-size *packets.*

1. *Describe in pseudo-code a scheduling algorithm working in cell-mode.*
2. *Describe in pseudo-code a scheduling algorithm working in packet-mode.*
3. *Compare the performance of the cell-mode algorithm and the packet-mode algorithm in terms of throughput and delay.*

Solution:
The two pseudocodes are available in Fig.s 3.3-3.4.

```
// CELL-MODE SCHEDULER
// Q[i][j] is the number of cells in VOQ[i][j]
int matching[N] // m[i]=j if input i is connected to output j; else NOT_USED
int output_reserved[M] // TRUE/FALSE

// init
for (j=0; j<M; j++)
   output_reserved[j]=FALSE
for (i=0; i<N; i++)
   matching[i]=NOT_USED
// schedule
for (i=0; i<N; i++) // for each input
      for (j=0; j<M; j++) // for each output
         if (Q[i][j]>0 && output_reserved[j]==FALSE)
            // found available HoL HEAD cell
            matching[i]=j
            output_reserved[j]=TRUE
            break
```

Figure 3.3: Pseudocode for cell-mode scheduler in Ex. 123

```
// Q[i][j] is the number of cells in VOQ[i][j]
// S[i][j] is the state of head-of-line cell of VOQ[i][j]: HEAD/MIDDLE/TAIL
static int matching[N] // m[i]=j if in. i is connected to out. j; else NOT_USED
int output_reserved[M] // TRUE/FALSE

// init
for (j=0; j<M; j++)
   output_reserved[j]=FALSE
// check old matching
for (i=0; i<N; i++)
   j=matching[i]
   if (j>=0 && Q[i][j]>0 && S[i][j]!=HEAD)
      output_reserved[j]=TRUE // keep old edge
   else
      matching[i]=NOT_USED // for empty queues or HEAD-cells
// schedule
for (i=0; i<N; i++) // for each input
   if (matching[i]==NOT_USED) // check if the input is still available
      for (j=0; j<M; j++) // for each output
         if (Q[i][j]>0 && output_reserved[j]==FALSE)
            // found available HoL HEAD cell
            matching[i]=j
            output_reserved[j]=TRUE
            break
```

Figure 3.4: Pseudocode for packet-mode scheduler in Ex. 123

3.2.3 Scheduling algorithms for QoS support

Exercise 124 (out of scope for the program 2021/22)
Consider a 4×4 input queued switch, with each port running at 5 Gbps. The following rate matrix must be guaranteed:

$$\hat{R} = \begin{bmatrix} 0 & 1.5 & 1 & 2 \\ 1 & 1.5 & 0 & 1 \\ 2 & 0.5 & 1.5 & 0 \\ 1 & 0 & 1 & 1.5 \end{bmatrix} \textit{ Gbps}$$

- *Describe the scheduling algorithm.*
- *Show the frame sequence.*
- *Compute the minimum length of the frame.*

Matching and VOQs	Average delay	Maximum delay
$M_1: 1 \to 2,\ 2 \to 1,\ 3 \to 3,\ 4 \to 4$	1.2 slots=123 ns	3 slots=307 ns
$M_2: 1 \to 4,\ 2 \to 2,\ 3 \to 1,\ 4 \to 3$	1.2 slots=123 ns	3 slots=307 ns
$M_3: 1 \to 3,\ 2 \to 4,\ 3 \to 2,\ 4 \to 1$	2.0 slots=205 ns	4 slots=410 ns

Table 3.3: Access delays for frame $(M_1, M_1, M_2, M_2, M_3)$

Matching and VOQs	Average delay	Maximum delay
$M_1: 1 \to 2,\ 2 \to 1,\ 3 \to 3,\ 4 \to 4$	0.8 slots=82 ns	2 slots=205 ns
$M_2: 1 \to 4,\ 2 \to 2,\ 3 \to 1,\ 4 \to 3$	0.8 slots=82 ns	2 slots=205 ns
$M_3: 1 \to 3,\ 2 \to 4,\ 3 \to 2,\ 4 \to 1$	2.0 slots=205 ns	4 slots=410 ns

Table 3.4: Access delays for frame $(M_1, M_2, M_1, M_2, M_3)$

- *Compute the corresponding maximum and average* access delay *(i.e. under low traffic load), for each input-output couple, assuming that all the packets are 64 bytes long.*
- *Does the choice of the sequence of services in the frame affect the performance (throughput and delay)?*

Solution: From $\hat{R}$, the normalized rate matrix is:

$$R = \begin{bmatrix} 0 & 0.3 & 0.2 & 0.4 \\ 0.2 & 0.3 & 0 & 0.2 \\ 0.4 & 0.1 & 0.3 & 0 \\ 0.2 & 0 & 0.2 & 0.3 \end{bmatrix}$$

Since R is sub-stochastic, a double stochastic matrix $\tilde{R}$ must be found such that: $\tilde{R} \geq R$. A possible solution is the following:

$$\tilde{R} = \begin{bmatrix} 0 & 0.4 & 0.2 & 0.4 \\ 0.4 & 0.4 & 0 & 0.2 \\ 0.4 & 0.2 & 0.4 & 0 \\ 0.2 & 0 & 0.4 & 0.4 \end{bmatrix}$$

The Birkhoff-von Neumann (BvN) algorithm gives the following decomposition:

$$\tilde{R} = 0.4 \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + 0.4 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + 0.2 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

from which:

$$\tilde{R} = 0.4M_1 + 0.4M_2 + 0.2M_3$$

The timeslot is $T = 102.4$ ns. If the frame size is 5 (equal to 512 ns), the required rates are achievable through the following frame: $(M_1, M_1, M_2, M_2, M_3)$ from which the access delay for all possible input-output pairs (VOQs) are shown in table 3.3. Consider now another frame: $(M_1, M_2, M_1, M_2, M_3)$. Table 3.4 shows the corresponding access delays. Observe that in this second case, the access delay for all the VOQs corresponding to M_1 and M_2 decreases. However, the worst case access delay, due to the VOQs in M_3, does not change.

Hence, the temporal positions of the matchings inside the frame affect in general the delays but not the throughput, which is guaranteed to satisfy the rate matrix.

★★★

Exercise 125 (out of scope for the program 2021/22)
Consider a 5×5 input queued switch, with each port running at 10 Gbps. The following rate matrix must be guaranteed:

$$\hat{R} = \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 3 \\ 2 & 0 & 2 & 1 & 1 \\ 1 & 1 & 1 & 4 & 0 \\ 2 & 3 & 1 & 2 & 2 \end{bmatrix} \text{ Gbps}$$

1. *Describe the scheduling algorithm.*
2. *Show the frame sequence.*
3. *Compute the minimum length of the frame.*
4. *Compute the maximum access delay, under low traffic load, for each input-output couple, assuming that all the packets are 64 bytes long.*
5. *Does the choice of the sequence of services in the frame affect the performance (throughput and delay)?*

Solution: The exercise is identical to ex. 124, but with different parameters.

★★★

Exercise 126 (out of scope for the program 2021/22)
Consider a 5×5 input queued switch, with ports running at 10 Gbps. The following rates should be guaranteed:

$$R = \begin{bmatrix} 1 & 2 & 4 & 2 & 0 \\ 2 & 1 & 0 & 3 & 1 \\ 0 & 1 & 3 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 1 & 3 & 1 \end{bmatrix} \text{ Gbps}$$

where R_{ij} is the rate from input i to output j. Show how to guarantee these rates and describe all the algorithms involved.

★★★

Exercise 127 (out of scope for the program 2021/22)
Consider a 3×3 input queued switch, with ports running at 10 Gbps. The following rates should be guaranteed:

$$R = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 0 \\ 0 & 1 & 3 \end{bmatrix} \text{ Gbps}$$

where R_{ij} is the rate from input i to output j. Show how to guarantee these rates and describe all the algorithms involved.

★★★

Exercise 128 (out of scope for the program 2021/22)
Consider a 3×3 input queued switch, with ports running at 8 Gbps. Assume that the internal timeslot corresponds to a 64 bytes packet. The following rates should be guaranteed:

$$R = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix} \text{ Gbps}$$

where R_{ij} is the rate from input i to output j.

1. *Use the Paul algorithm to find a possible frame sequence F:*
   - *Show the corresponding Clos network*
   - *Show the Paul matrix evolution*
   - *Show the final F*
2. *Compute the average access delay (in nanoseconds) for each VOQ when adopting F*
3. *Compute the admissibility conditions for the traffic when adopting F*

Solution:

The Clos network is built with 3 I-stage modules 7×7, 7 II-stage modules 3×3 and 3 III-stage modules 7×7. We must connect R'_{ij} circuits from I-stage module i to III-stage module j and the corresponding request matrix $R' = [R'_{ij}]$ will be:

$$R' = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$$

We do not report here the steps of the Paul algorithm. One possible solution would be a frame with 7 matchings: $\mathcal{F} = [M_i]_{i=1}^{7}$, being:

$$M_1 = M_2 = M_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad M_4 = M_5 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad M_6 = M_7 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The access delays are the following:

VOQ	Serving matchings	Average access delay (in timeslots)
11, 22, 33	M_1, M_2, M_3	(1+1+1+5+4+3+2)/7=17/7
12, 23, 31	M_4, M_5	(4+3+2+1+1+6+5)/7=22/7
13, 21, 32	M_6, M_7	(6+5+4+3+2+1+1)/7=22/7

The absolute delay can be computed by multiplying the last column for the duration T of each timeslot, being

$$T = \frac{64 \times 8 \text{ bit}}{8 \text{ Gbps}} = 64 \text{ ns}$$

When adopting the above frame $\mathcal{F}$, in terms of normalized arrival rates, the admissibility conditions will be

$$\Lambda' \leq R'/7$$

In terms of absolute arrival rates, the admissibility conditions will be:

$$\Lambda = R'/7 \times 8 \text{ Gbps} = \begin{bmatrix} 3.43 & 2.28 & 2.28 \\ 2.28 & 3.43 & 2.28 \\ 2.28 & 2.28 & 3.43 \end{bmatrix} \text{ Gbps}$$

Exercise 129 (out of scope for the program 2021/22)

Consider a 4×4 input queued switch with virtual output queues (VOQs), with each port running at 10 Gbps. Assume that the internal timeslot corresponds to a 64 bytes packet. The following rate matrix must be guaranteed:

$$\hat{R} = \begin{bmatrix} 1 & 2 & 1 & 4 \\ 2 & 4 & 1 & 1 \\ 4 & 1 & 2 & 1 \\ 1 & 1 & 4 & 2 \end{bmatrix} \textit{ Gbps}$$

1. *Find the frame sequence F, using Paul algorithm.*
2. *What are the admissibility conditions for the traffic to achieve the maximum throughput?*
3. *What is the minimum worst-case access delay and the corresponding VOQs?*
4. *What is the maximum worst-case access delay and the corresponding VOQs?*

Solution:

1. Using Paul algorithm, a possible frame of 8 timeslots $F = [M_i]_{i=1}^{8}$ is the following:

$$M_1 = M_2 = M_3 = M_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad M_5 = M_6 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_7 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad M_8 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

2. The traffic matrix Λ to be admissile and achieve the maximum throughput must satisfy:

$$\Lambda \leq \hat{R}/8 \times 10 = \begin{bmatrix} 1.25 & 2.50 & 1.25 & 5.00 \\ 2.50 & 5.00 & 1.25 & 1.25 \\ 5.00 & 1.25 & 2.50 & 1.25 \\ 1.25 & 1.25 & 5.00 & 2.50 \end{bmatrix} \text{ Gbps}$$

3. According to F, all the VOQs corresponding to M_1 experience a worst-case access delay equal to 5 timeslots, which is the minimum among all the VOQs.
4. According to F, all the VOQs corresponding to M_7 and M_8 experience a worst-case access delay equal to 8 timeslots, which is the maximum among all the VOQs.

★★★

Exercise 130 (out of scope for the program 2021/22)
Consider a 4×4 input queued switch, with each port running at 8 Gbps. Assume that the internal timeslot corresponds to a 64 bytes packet. The following rate matrix must be guaranteed:

$$\hat{R} = \begin{bmatrix} 0 & 1 & 4 & 2 \\ 1 & 2 & 0 & 1 \\ 4 & 0 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix} \textit{ Gbps}$$

1. *Use the Paul algorithm to find the possible frame sequence, named F_1.*
2. *Use the Birkhoff-von Neumann decomposition to find the possible frame sequence, named F_2.*
3. *Are F_1 and F_2 the same? Why?*
4. *Under which admissibility conditions, the two frame sequences F_1 and F_2 allow to obtain the maximum throughput?*

5. *Rearrange F_1 and F_2 to maximize the worst case access delay, under low traffic load, for the flow from input 1 to output 3. Compute this delay in μs.*

6. *Rearrange F_1 and F_2 to minimize the worst case access delay, under low traffic load, for the flow from input 1 to output 3. Compute this delay in μs.*

★★★

Exercise 131 (out of scope for the program 2021/22)
Consider a 4×4 input queued switch, with each port running at 8 Gbps. Assume that the internal timeslot corresponds to a 64 bytes packet. The following rate matrix must be guaranteed:

$$\hat{R} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 0 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 0 & 2 \end{bmatrix} \text{ Gbps}$$

1. *Draw the simplest Clos network which allows to use Paul algorithm to decompose the matrix.*

2. *Find the possible frame sequence, named F_1, according to Paul algorithm.*

3. *Use the Birkhoff-von Neumann decomposition to find the possible frame sequence, named F_2.*

4. *Are F_1 and F_2 the same? Why?*

5. *Under which admissibility conditions, the two frame sequences F_1 and F_2 allow to obtain the maximum throughput?*

★★★

Exercise 132 (out of scope for the program 2021/22)
Consider a 4×4 input queued switch, with ports running at 10 Gbps. The following rates should be guaranteed:

$$R = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 2 & 1 & 0 & 4 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ Gbps}$$

where R_{ij} is the rate from input i to output j. Show how to guarantee these rates and describe all the algorithms involved.

★★★

Exercise 133 (out of scope for the program 2021/22)
Consider a slotted 4×4 input-queued switch, with input ports running at 100 Mbit/s. Each slot lasts 5 μs. The following rate matrix must be guaranteed:

$$\Lambda = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} Mbit/s$$

Answer the following:

1. *Is the traffic admissible? Why?*

2. *Compute a matching sequence in a frame F_1 to support Λ, and to minimize the frame duration and the worst case access delay.*

3. *Compute a matching sequence in a frame F_2 to support Λ with only 4 matchings.*

For each frame F_1 and F_2,

1. *What is the frame duration in [s]?*

2. *What is the worst case access delay for each input-output couple?*

Solution: The traffic is admissible, since the highest load on a input/output port is 58 Mbit/s, which is less than 100 Mbit/s.

To minimize the frame duration, Λ can be augmented (simply) up to:

$$\Lambda' = \begin{pmatrix} 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 \end{pmatrix} \ Mbit/s$$

for which the frame is trivially:

$$F_1 = (M_1, M_2, M_3, M_4)$$

where M_k is the matching corresponding to the i-th generalized diagonal in the matrix. Now F_1 lasts 4 slots, i.e. 20 μs; the corresponding worst case access delay is 3 slots, i.e. 15 μs and the average access delay is 1.5 slots, i.e. 7.5 μs.

Note that F_1 satisfies also the requirements for F_2.

★★★

Exercise 134 (out of scope for the program 2021/22)
Consider a generic frame scheduling policy for input queued switches in which the arrival rates are known and the scheduling decision are taken oblivious of the queue state.

1. *Prove that the average delay grows as $O(N)$ in order sense, for one of the possible frame scheduling algorithms.*

2. *Is it possible to improve the average delay in order sense? How?*

If needed, for a slotted Geom/Geom/1 queue with arrival probability λ and service probability μ, the average delay is:

$$\overline{W}_{Geom/Geom/1} = \frac{\eta}{\lambda(1-\eta)} \qquad \textit{with} \qquad \eta = \frac{\lambda(1-\mu)}{\mu(1-\lambda)}$$

If needed, for a slotted M/D/1 queue with binomial $(N, \rho/N)$ arrivals per slot, the average delay is:

$$\overline{W}_{M/D/1} = 1 + \frac{\rho}{2(1-\rho)}$$

3.2.4 Scheduling algorithms for multicast traffic

Exercise 135
Describe in pseudo-code an algorithm to schedule the transmissions of unicast and multicast packets in a slotted input queued switch, of size $N \times P$. Assume that the queue structure at each input port is the following: i) a single FIFO queue for all unicast packets destined to a particular output; ii) a single FIFO queue for all the multicast packets. At each timeslot, let

```
// initialize the data structures
for j=1...P     // for each output port
    output_reserved[j]=false
    for i=1...N // for each input port
        X[i][j]=false
// scheduler decision
for i=1...N     // for each input port
    // try to serve the multicast traffic
    multicast_served=false
    if (M[i]>0) // check if the mc queue is non-empty
       for j=1...P // for each output port
           if (destInMCQueue(j,i) AND (output_reserved[j]==false))
              // it means that the output is present in the
              // fanout set and it has not been reserved so far
              output_reserved[j]=true
              X[i][j]=true
              multicast_served=true
   // try to serve a unicast queue only if it is available
   if (multicast_served==false)
      // M[i] has not served; now look at unicast traffic
      for j=1...P
          if ((Q[i][j]>0) AND (output_reserved[j]==false))
             X[i][j]=true
             output_reserved[j]=true
             break
```

Figure 3.5: Pseudocode for Ex. 135

`Q[i][j]` be the size of the queue for unicast packets at input i and destined to output j. Let `M[i]` be the size of the queue for multicast packets at input i. Let X a matrix describing the switching configuration chosen in the current timeslot, based on the state of the queues. More precisely, `X[i][j]` is a boolean variable, which assumes the value true iff the crosspoint from input i to output j is active, i.e. a packet is sent from input "i to output j in the current timeslot.

1. *Compute the total number of queues for each input and in the whole switch.*
2. *Write in pseudo-code a maximal scheduling algorithm that allows fanout splitting and serves the multicast packets at higher priority with respect to unicast packets. Assume that a function `destInMCQueue(j,i)` that returns true iff output j belongs to the fanout set of the packet at the head of the multicast queue `M[i]` at input i is available.*
3. *Does the algorithm achieve 100% throughput under any admissible traffic? Why? (question out of scope for the program 2021/22)*

Solution: The total number of queues for each input is $P + 1$, and for the whole switch is $N(P + 1)$.

The psedocode is reported in Fig. 3.5.

The switch cannot obtain the maximum throughput because of any of the following reasons: (i) the queueing is not optimal and suffers the HoL blocking problem for multicast traffic, (ii) the scheduling policy is not optimal. In general, an input queued switch cannot obtain the maximum throughput under any admissible multicast traffic because of intrinsic architecture limitations, highlighted by specific arrivals patterns.

★★★

Exercise 136

Consider an input queued switch of size $N \times M$ supporting both unicast and multicast traffic. Assume that each input is equipped with just 2 queues: one queue for broadcast packets and one queue for multicast (but not broadcast) traffic, including unicast traffic.

Consider a scheduling algorithm that does not allow fanout-splitting and serves broadcast traffic at highest priority. As a reminder, the fanout set of a multicast packet is the set of its destination ports.

1. *Show an example (if any) of admissible arrival pattern for which the scheduler achieves the maximum throughput. (question out of scope for the program 2021/22)*
2. *Show an example (if any) of admissible arrival pattern for which the scheduler does not achieve the maximum throughput. (question out of scope for the program 2021/22)*
3. *What are the performances of the switch fed by unicast traffic only? (question out of scope for the program 2021/22)*
4. *Describe in pseudo-code the scheduling algorithm, using the notation below.*

At each timeslot, let `B[i]` *be the size of the queue for broadcast packets at input* i*. Let* `M[i]` *be the size of the queue for multicast/unicast packets at input* i*. Assume that function* `destInMCQueue(j,i)` *returns* `true` *iff output* j *belongs to the fanout set of the packet at the head of the multicast queue* `M[i]`*. Let* `X` *be the matrix describing the switching configuration chosen in the current timeslot, based on the state of the queues. More precisely,* `X[i][j]` *is a boolean variable, which assumes the value* `true` *iff the crosspoint from input* i *to output* j *is active, i.e. a packet must be sent from input* i *to output* j *in the current timeslot.*

Solution:

The switch is able to achieve the maximum throughput when all the inputs are receiving broadcast packets with probability $\leq 1/N$ each.

The switch is not able to achieve the maximum throughput for generic multicast packets, due to the no-fanout splitting policy. For example, consider input 1 receiving packets with fanout set $\{1, 2\}$ (with prob. 0.5 per timeslot) and $\{3, 4\}$ (with prob. 0.5 per timeslot) and input 2 receiving packets with fanout set $\{1, 3\}$ (with prob. 0.5 per timeslot) and $\{2, 4\}$ (with prob. 0.5 per timeslot). Even if the traffic is admissible, at most one packet is served per timeslot; hence, half of the packets are lost.

Under unicast traffic only, the switch behaves exactly as a single queue per input switch. Under admissible uniform i.i.d. Bernoulli arrivals, the maximum throughput is around 58%.

The pseudocode is available in Fig. 3.6.

★★★

Exercise 137

Consider an input queued switch of size $N \times M$ *supporting both unicast and multicast traffic. Assume that each input is equipped with just one queue. The scheduling algorithm is designed (i) to allow fanout-splitting, and (ii) to serve always the packet with largest fanout among all the inputs, at each timeslot. As a reminder, the fanout set of a multicast packet is the set of its destination ports.*

1. *Define properly the concept of "admissible multicast traffic", with the help of formulas. (question out of scope for the program 2021/22)*
2. *Show an example (if any) of admissible arrival pattern for which the scheduler achieves the maximum throughput. (question out of scope for the program 2021/22)*
3. *Show an example (if any) of admissible arrival pattern for which the scheduler does not achieve the maximum throughput. (question out of scope for the program 2021/22)*
4. *Describe in pseudo-code the scheduling algorithm, using the notation below.*

```
// initialize the data structures
for j=1...M     // for each output port
    output_reserved[j]=false
    for i=1...N // for each input port
        X[i][j]=false
// first, try to serve broadcast packets
for i=1...N     // for each input port
    if (B[i]>0)
       // found broadcast packet with all available outputs
       for j=1...N
           X[i][j]=true
           output_reserved[j]=true // (not needed)
       return // ends since switching configuration is maximal
// second, try to serve multicast packets
for i=1...N
    if (M[i]>0) // check if the mc queue is non-empty
       // check if all the corresponding outputs are available
       multicast_available=true
       for j=1...M // for each output port
           if (destInMCQueue(j,i))
              // j is in the fanout set of the packet
              if (output_reserved[j])
                 // the output is already reserved
                 multicast_available=false
                 break // useless to continue to check
       if (multicast_available)
          // reserve all the outputs
          for j=1...M // for each output port
              if (destInMCQueue(j,i))
                 output_reserved[j]=true
                 X[i][j]=true
```

Figure 3.6: Pseudocode for Ex. 136

At each timeslot, let `M[i]` *be the size of the queue at input* i*. Function* `destInMCQueue(j,i)` *returns* `true` *iff output* j *belongs to the fanout set of the packet at the head of the multicast queue* `M[i]`*. Let* `X` *be a boolean matrix describing the switching configuration chosen in the current timeslot, based on the state of the queues:* `X[i][j]` *is* `true` *iff the crosspoint from input* i *to output* j *is active, i.e. a packet must be sent from input* i *to output* j *in the current timeslot.*

Solution: The pseudocode is available in Fig. 3.7.

★★★

Exercise 138
Consider an input queued switch of size $N \times M$ *supporting both unicast and multicast traffic. Assume that each input is equipped with just* 2 *queues: one queue for broadcast packets and one queue for both multicast and unicast traffic.*

Consider a scheduling algorithm that does not allow fanout-splitting *and serves broadcast traffic at highest priority.*

1. *Show an example (if any) of admissible arrival pattern for which the scheduler achieves the maximum throughput. (question out of scope for the program 2021/22)*
2. *Show an example (if any) of admissible arrival pattern for which the scheduler does not achieve the maximum throughput. (question out of scope for the program 2021/22)*
3. *What are the performances of the switch fed by unicast traffic only? (question out of scope for the program 2021/22)*
4. *Describe in pseudo-code the scheduling algorithm, using the notation below.*

```
// initialize the data structures
for j=1...M     // for each output port
    output_reserved[j]=false
    for i=1...N // for each input port
        X[i][j]=false
// first, find the largest fanout
max_fanout=0;
max_fanout_input=-1;
for i=1...N     // for each input port
    // skip empty queues
    if M[i]==0
       continue // consider another input
    // compute the fanout
    acc=0
    for j=1...M // check if each output is in the fanout set
         if (destInMCQueue(j,i))
            acc++
    // now acc is the fanout, update eventually the max
    if (acc>max_fanout)
        max_fanout=acc
        max_fanout_input=i
// reserve the largest fanout (if available)
if (max_fanout>0)
    for j=1...M
         if (destInMCQueue(j,max_fanout_input))
            X[max_fanout_input][j]=true // reserve crosspoint
            output_reserved[j]=true // reserve output
// second, serve all the other inputs in a greedy fashion
for i=1...N
     // skip the input already reserved
     if (i==max_fanout_input)
          continue // consider another input
     if (M[i]>0)
          // found a non empty queue
          for j=1...M
               if (destInMCQueue(j,i) AND (!output_reserved[j]))
                   X[i][j]=true // reserve crosspoint
                   output_reserved[j]=true // reserve output
```

Figure 3.7: Pseudocode for Ex. 137

At each timeslot, let `B[i]` *be the size of the queue for broadcast packets at input* i*. Let* `M[i]` *be the size of the queue for multicast/unicast packets at input* i*. Assume that function* `destInMCQueue(j,i)` *returns* `true` *iff output* j *belongs to the fanout set of the packet at the head of the multicast queue* `M[i]`*. Let* X *be the matrix describing the switching configuration chosen in the current timeslot, based on the state of the queues. More precisely,* `X[i][j]` *is a boolean variable, which assumes the value* `true` *iff the crosspoint from input* i *to output* j *is active, i.e. a packet must be sent from input* i *to output* j *in the current timeslot.*

Solution:

The switch is able to achieve the maximum throughput when all the inputs are receiving broadcast packets with probability $\leq 1/N$ each.

The switch is not able to achieve the maximum throughput for generic multicast packets, due to the no-fanout splitting policy. For example, consider input 1 receiving packets with fanout set $\{1, 2\}$ (with prob. 0.5 per timeslot) and $\{3, 4\}$ (with prob. 0.5 per timeslot) and input 2 receiving packets with fanout set $\{1, 3\}$ (with prob. 0.5 per timeslot) and $\{2, 4\}$ (with prob. 0.5 per timeslot). Even if the traffic is admissible, at most one packet is served per timeslot; hence, half of the packets are lost.

Under unicast traffic only, the switch behaves exactly as a single queue per input switch. Under admissible uniform i.i.d. Bernoulli arrivals, the maximum throughput is around 58%.

The pseudocode is available in Fig. 3.8.

```
// initialize the data structures
for j=1...M     // for each output port
    output_reserved[j]=false
    for i=1...N // for each input port
        X[i][j]=false
// first, try to serve broadcast packets
for i=1...N     // for each input port
    if (B[i]>0)
       // found broadcast packet with all available outputs
       for j=1...N
           X[i][j]=true
           output_reserved[j]=true // (not needed)
       return // ends since switching configuration is maximal
// second, try to serve multicast packets
for i=1...N
    if (M[i]>0) // check if the mc queue is non-empty
       // check if all the corresponding outputs are available
       multicast_available=true
       for j=1...M // for each output port
           if (destInMCQueue(j,i))
              // j is in the fanout set of the packet
              if (output_reserved[j])
                 // the output is already reserved
                 multicast_available=false
                 break // useless to continue to check
       if (multicast_available)
          // reserve all the outputs
          for j=1...M // for each output port
              if (destInMCQueue(j,i))
                 output_reserved[j]=true
                 X[i][j]=true
```

Figure 3.8: Pseudocode for Ex. 138

★★★

Exercise 139
Consider an input queued switch of size $N \times M$ supporting both unicast and multicast traffic. Assume that each input is equipped with just 2 queues: one queue for broadcast packets and one queue for multicast (but not broadcast) traffic, including unicast traffic. Consider a scheduling algorithm that allows fanout-splitting and serves broadcast traffic at highest priority. As a reminder, the fanout set of a multicast packet is the set of its destination ports.

Describe in pseudo-code the scheduling algorithm, using the following notation. At each timeslot, let `B[i]` be the size of the queue for broadcast packets at input i. Let `M[i]` be the size of the queue for multicast/unicast packets at input i. Assume that function `destInMCQueue(j,i)` returns `true` iff output j belongs to the fanout set of the packet at the head of the multicast queue `M[i]`. Let `X` be the matrix describing the switching configuration chosen in the current timeslot, based on the state of the queues. More precisely, `X[i][j]` is a Boolean variable, which assumes the value `true` iff the crosspoint from input i to output j is active, i.e. a packet must be sent from input i to output j in the current timeslot.

Solution: The pseudocode is available in Fig. 3.9.

★★★

Exercise 140 (out of scope for the program 2021/22)
Consider an $N \times M$ input queued switch supporting multicast traffic, with optimal queueing structure MC-VOQ for multicast.

1. *Define the MC-VOQ queueing and compute the total number of queues.*

```
// initialize the data structures
for j=1...M     // for each output port
    output_reserved[j]=false
    for i=1...N // for each input port
        X[i][j]=false
// first, try to serve broadcast packets
for i=1...N     // for each input port
    if (B[i]>0)
       // found broadcast packet with all available outputs
       for j=1...N
           X[i][j]=true
           output_reserved[j]=true // (not needed)
       return // ends since switching configuration is maximal
// second, try to serve multicast packets
for i=1...N
    if (M[i]>0) // check if the mc queue is non-empty
       for j=1...M // for each output port
           // check is any output in the fanout set is available
           if (destInMCQueue(j,i)==true AND output_reserved[j]==false)
                 output_reserved[j]=true
                 X[i][j]=true
```

Figure 3.9: Pseudocode for Ex. 139

2. *Write the pseudocode of a maximal scheduler allowing fanout-splitting (i.e. a multicast packet can be sent to a subset of destinations). Comment the code. Use the data structures reported below. Any data structure must be properly defined and eventually initialized.*

3. *Comment about the fairness of the above code.*

4. *In which application scenarios multicast traffic is relevant for high speed networking?*

As notation, assume that `Q[i][k]` *is the occupancy of queue* `k` *at input* `i` *and* `FS(k)` *returns the fanout set (i.e. the set of destination outputs) of the packets stored in queue* `k`*. The crossbar configuration is described by two vectors: let* `X` *be a vector of size* M *defined as follows:* `X[j]=i` *whenever input* `i` *sends a copy of the packet to output* `j`*; let* `Y` *be a vector of size* N *describing the queues to serve, defined as* `Y[i]=k` *whenever queue* `k` *is served at input* `i`*.*

Solution:

1. In the MC-VOQ structure, each input has one dedicated queue for each possible fanout set, thus a total of $N \times (2^M - 1)$ queues are present in the switch.

2. See below code.

3. The below code is unfair since the queues belonging to the first inputs and the ones identified by smaller values of k will be served at higher priority than the other queues.

4. Example of applications for which multicast is important is: broadcast in VLAN scenarios or virtualized networks, duplication of data in data center, financial transactions (high-frequency trading).

```
// initialize data structures
for j=1...M // for each output port
    X[j]=-1 // output j non connected
for i=1...N // for each input port
    Y[i]=-1 // the queue has not be chosen so far
// take scheduling decision
for i=1...N     // for each input port
     for k=1...(2^M-1) // for each queue
          if (Q[i][k]>0) // check if the queue is not empty
                for each j in FS(k) // for each output in the fanout set
                       if (X[j]==-1) // check if the output is still available
                          // matching the input with the output
                          Y[i]=k // store the queue to serve
                          X[j]=i // match input i with output j
          if (Y[i]>0) // check if an input has been found
              break // consider a new input
```

Chapter 4

Fast packet classification and SDN

4.1 Lookup tables for packet forwarding

Exercise 141
Given the following routing table:

```
IP routing table
Destination/Netmask     Gateway (Rule)
32.0.0.0/4              A
32.0.0.0/6              B
40.0.0.0/6              C
56.0.0.0/5              D
56.0.0.0/8              E
57.0.0.0/9              F
57.128.0.0/9            G
default                 H
```

1. *Build the corresponding binary trie.*
2. *Build the corresponding patricia trie.*

Solution: Compute the prefix list:

```
Prefix  Rule
0010*           A
001000*         B
001010*         C
00111*          D
00111000*       E
001110010*      F
001110011*      G
*               H
```

Fig.s 4.1 and 4.2 show the binary trie and the patricia trie.

★★★

Exercise 142
Given the following routing table:

```
IP routing table
Destination/Netmask     Gateway (Rule)
64.0.0.0/3              A
64.0.0.0/5              B
80.0.0.0/5              C
112.0.0.0/4             D
112.0.0.0/7             E
114.0.0.0/8             F
115.0.0.0/8             G
default                 H
```

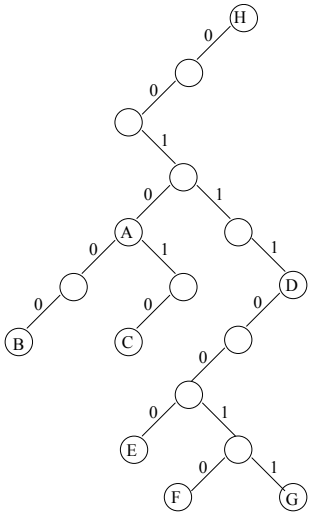


Figure 4.1: Binary trie (Ex. 141)

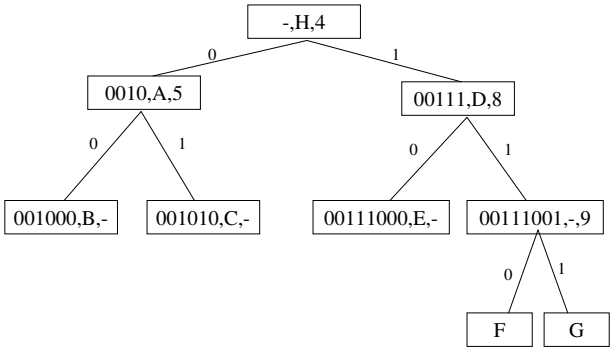


Figure 4.2: Patricia trie, in the format: “prefix, rule, next-bit-to-inspect” (Ex. 141)

1. *Build the corresponding binary trie.*
2. *Build the corresponding patricia trie.*

Solution:

The exercise is identical to Ex. 141 but with an initial “0” in the prefix. Thus the solution can be easily derived from Fig.s 4.1 and 4.2.

★★★

Exercise 143

Given the following routing table:

```
IP routing table
Destination/Netmask     Gateway (Rule)
64.0.0.0/2              B
96.0.0.0/4              C
112.0.0.0/4             D
112.0.0.0/5             M
120.0.0.0/8             L
128.0.0.0/1             E
192.0.0.0/3             F
224.0.0.0/4             G
238.0.0.0/7             H
239.128.0.0/9           I
default                 A
```

1. *Build the corresponding binary trie.*
2. *Build the corresponding patricia trie.*
3. *Which trie is better than the other? Why?*

★★★

Exercise 144
Given the following routing table, represented as (prefix, rule):

$(0000, A), (0001, B), (001, C), (01, D), (010, E), (011, F), (1, G), (1000, H), (1001, I), (1110, L), (1111, M)$

1. *Build the corresponding binary trie.*
2. *Build the corresponding Patricia trie.*
3. *Evaluate the number of memory accesses when searching for a given address:*

Address	*Num. memory accesses*	
	Binary trie	*Patricia trie*
0000 0000		
0111 1111		
1011 1111		
1100 1100		
1111 1111		

★★★

Exercise 145
Given the following routing table:

```
IP routing table
Destination/Netmask     Gateway (Rule)
16.0.0.0/5              A
28.0.0.0/6              B
26.0.0.0/7              C
128.0.0.0/5             D
193.0.0.0/8             E
129.0.0.0/8             F
default                 G
```

1. *Build the corresponding binary trie, after having defined the fields within each node.*
2. *Build the corresponding Patricia trie, after having defined the fields within each node.*

Solution:
The corresponding binary trie and Patricia trie are shown in Figs. 4.3 and 4.4, respectively.

Exercise 146
Consider a packet classification mechanism for TCP/IP traffic, implemented in a router. Let us assume that the classification is based just on the destination IP address and on the TCP destination port. Build a compressed trie (based on Patricia trie) from the set of rules in Table 4.1. In addition, evaluate the minimum number of bits needed to represent all the trie, assuming that each pointer inside the trie requires 8 bits.

★★★

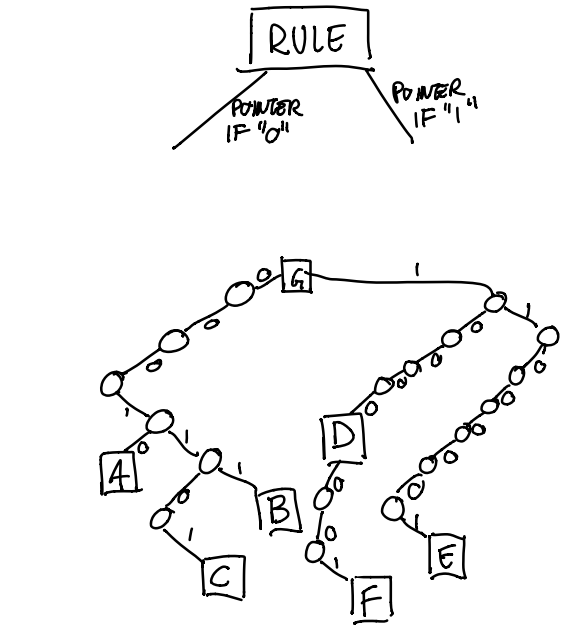


Figure 4.3: Binary trie (Ex. 145).

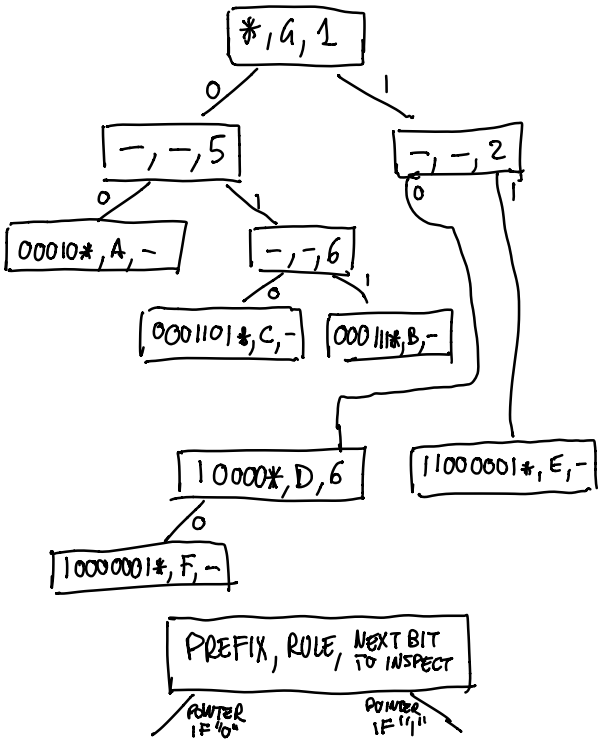


Figure 4.4: Patricia trie (Ex. 145).

Exercise 147

Consider the data structure for fast IP lookup based on (i) binary trie and (ii) Patricia trie. For simplicity, assume that IP addresses are coded on 4 bits, instead of the usual 32 bits.

Table 4.1: Set of rules (Ex. 146)

Destination IP address	Destination TCP port	Action
10*	80	A
0111*	25	B
100*	25	C
0111*	21	D
10*	25	E
1*	21	F
*	80	G
011*	25	H
0111*	80	I
010100*	25	L
*	25	M
*	*	N

1. *Show a routing table in which the binary trie has the same lookup time than the Patricia trie, and draw the corresponding trie;*
2. *Show a routing table in which the binary trie has a worst lookup time than the Patricia trie, and draw the corresponding trie.*

Assume that the lookup complexity is given simply by the number of comparison while visiting the trie.

Solution:

1.

Prefix	Rule
*	A

2.

Prefix	Rule
0000	A

4.2 Software Defined Networking

Exercise 148
Consider the new network paradigm denoted as "Software Defined Networking" (SDN).

1. *What is the main difference with the traditional Internet architecture?*
2. *What is a SDN controller? How can its API interfaces be classified?*
3. *What is Openflow?*
4. *How is a flow table defined in Openflow?*
5. *What are the main messages defined in Openflow and what are their purposes?*
6. *What are the main consequences of Openflow on the design of the switching architectures?*

★★★

Exercise 149
Consider the new network paradigm denoted as "Software Defined Networking" (SDN).

1. *What are the flow tables?*
2. *Show three examples of flow rules corresponding to (i) an Ethernet switch, (ii) an IP router and (iii) a firewall.*
3. *Explain the differences between Single Match Tables, Multiple Match Tables and Reconfigurable Match Tables.*
4. *Show an example in which the three kind of tables are different for the same set of flow rules.*
5. *What is the difference between Openflow and P4, referring to the flow tables?*

★★★

Exercise 150
In a Software-Defined-Network (SDN), any switch is equipped with a match-action *table that is followed to process and forward incoming data. Describe a software-based, compressed data structure to store efficiently the match-action rules reported in Table 4.2. Describe an hardware-based solution based on a TCAM and evaluate the required memory size.*

Table 4.2: Match-action table (Ex. 150)

MAC source	MAC dest.	IP source	IP dest.	TCP port source	TCP port dest.	ACTION
*	*	01*	*	*	*	drop
M1	*	*	*	*	*	drop
*	*	*	1*	*	*	forward to output 1
*	*	*	101*	*	*	forward to output 1
*	*	*	10000*	*	*	forward to output 2
*	*	*	100001*	*	*	forward to output 1
*	*	*	100001*	*	80	drop
*	*	*	100001*	*	25	forward to output 3
*	*	*	01*	*	25	flood
*	*	*	*	*	*	drop

Solution:
Regarding the software implementation, a simple patricia trie built with just the rules in "IP destination" would be enough for the main structure of the decision tree. The patricia trie must be equipped with additional pointers to manage the whole set of rules.
Regarding the TCAM implementation, each row (rule) requires: $48 + 48 + 32 + 32 + 16 + 16$ *bits. In total, 10 rows are enough. An additional table with the 10 actions must be provided. Since each action can be coded with 3 bits, a minimum of 30 bits would be required.*

★★★

Exercise 151
Consider a SDN network based on Openflow as protocol adopted in the southbound interface.

1. *Describe all the three possible kinds of Openflow messages, highlighting the purpose of each of them.*
2. *What is a flow table in Openflow?*
3. *Describe the internal architecture of an Openflow switch, highlighting all the main components and the kinds of adopted memories.*

4. *Consider the scenario of Fig. 4.5 in which an Openflow switch is used to connect two hosts. Assume that Host 1 sends two Ethernet packets to Host 2, denoted as p_1 and p_2. Assume that the controller knows the network topology and adopts a reactive routing application, i.e. the flow tables are populated in realtime as soon as the traffic is received.*

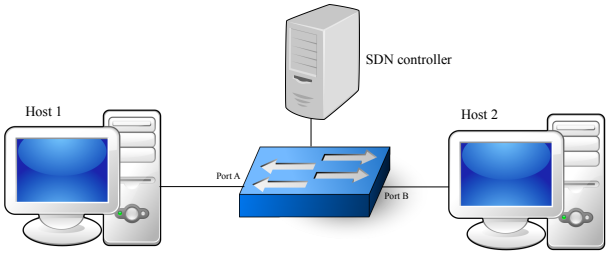


Figure 4.5: Network scenario (Ex. 151)

   (a) *Show the sequence of the packets that are exchanged in the data plane (between the hosts and the switch) and in the control plane (between the switch and the controller) by filling the following table:*

Sequence number	*Link*	*Packet*

   (b) *Show the final flow table in the switch*

Solution:

For the considered scenario, a possible sequence of packets is the following:

Seq.	Link	Packet
1	Host 1 $\rightarrow$ Switch	p_1
2	Switch $\rightarrow$ Controller	Packet-in(p_1, from port A)
3	Controller $\rightarrow$ Switch	Packet-out(p_1, to port B)
4	Controller $\rightarrow$ Switch	Flow-mod(send to port B all packets destined to MAC(Host 2))
5	Switch $\rightarrow$ Host 2	p_1
6	Host 1 $\rightarrow$ Switch	p_2
7	Switch $\rightarrow$ Host 2	p_2

The final flow table will be:

MAC-source	MAC-dest	IP-source	IP-dest	$\cdots$	Action
*	MAC(Host 2)	*	*	*	Send to port B

Exercise 152

Consider the network shown in Fig. 4.6, connecting host H1 to host H2 through 2 Openflow switches (A and B).

Assume now that two Ethernet packets are sent from H1 to H2 and that all the flow tables in the switches are initially empty. Show the sequence of packets observed in all the links (both data plane and control plane), highlighting the role of pkt-in, pkt-out and flow-mod messages. Show the final flow tables in both switches.

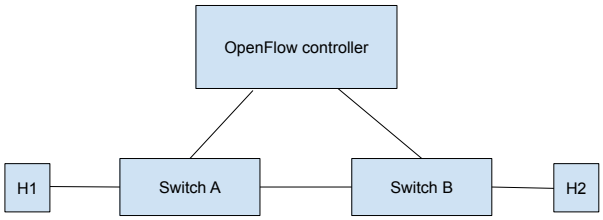


Figure 4.6: Openflow network scenario (Ex. 152)

Chapter 5

Probabilistic data structures

5.1 Hash tables and fingerprinting

Exercise 153
Consider a hash table with 4 buckets storing IP addresses and the corresponding number of packets.

1. *What is the difference between a hash function and a hash table?*
2. *Define a proper hash function.*
3. *According to the above hash function, describe the state of the hash table after each operation:*

Operation	*Bucket 1*	*Bucket 2*	*Bucket 3*	*Bucket 4*
Insert (10.0.0.1,10)				
Insert (10.0.1.1,15)				
Insert (10.0.2.1,20)				
Insert (10.0.3.1,25)				
Insert (10.0.2.1,30)				
Insert (10.0.4.1,40)				
Delete (10.0.0.1)				

Solution:
Let $a.b.c.d$ be the IP address represented in decimal format. Now a simple hash function could be: $h(a.b.c.d) = (a + b + c + d) \mod 4$.

Operation	Bucket 1	Bucket 2	Bucket 3	Bucket 4
Insert (10.0.0.1,10)				(10.0.0.1,10)
Insert (10.0.1.1,15)	(10.0.1.1,15)			(10.0.0.1,10)
Insert (10.0.2.1,20)	(10.0.1.1,15)	(10.0.2.1,20)		(10.0.0.1,10)
Insert (10.0.3.1,25)	(10.0.1.1,15)	(10.0.2.1,20)	(10.0.3.1,25)	(10.0.0.1,10)
Insert (10.0.2.1,30)	(10.0.1.1,15)	(10.0.2.1,30)	(10.0.3.1,25)	(10.0.0.1,10)
Insert (10.0.4.1,40)	(10.0.1.1,15)	(10.0.2.1,30)	(10.0.3.1,25)	(10.0.0.1,10), (10.0.4.1,40)
Delete (10.0.0.1)	(10.0.1.1,15)	(10.0.2.1,30)	(10.0.3.1,25)	(10.0.4.1,40)

★★★

Exercise 154
Consider a traffic monitoring system that tracks the number of IP packets transferred for each traffic flow. A flow x is identified by the pair of IP source and destination addresses. We assume that the data structure to store the number of packets for each flow is a hash table with 65,536 buckets, specifically designed to exploit the "power of 2 random choices" result for random load balancing.

1. *What is the claim of the "power of 2 random choice" result? Why is it relevant for hash tables?*

2. *Describe in details two hash functions $h_1(x)$ and $h_2(x)$ specifically designed for the flow identifier considered in the problem and for the given size of the hash table.*

3. *Describe in pseudocode the operation to update the hash table when a new packet of flow x is processed.*

Solution:

1. See the class notes

2. Each hash function must map a sequence of 64 bits (i.e. 32 bits for the IP source and 32 bits for the IP destination), denoted as x to a sequence of 16 bits (i.e. $\log_2 65536$, to index each bucket of the hash table). Let x_1 the integer value ($\in [0, 2^{32} - 1]$) corresponding to the IP source address and the x_2 the one corresponding to the IP destination address. Thus, one possible option would be $h_1(x) = x_1 \mod 2^{16}$ and $h_2(x) = x_2 \mod 2^{16}$, i.e. $h_1(x)$ uses the 16 least significant bit of the source IP address and $h_2(x)$ the 16 least significant bit of true source IP address. Notably, using instead the most significant bits would perform poorly (i.e. higher hash collision) due to the particular structure of the IP addresses.

3. Let T be the hash table and $T[i]$ is the ith bucket, implementing a linked list to store elements in the format (x, v), where x is the flow identifier and v is the corresponding number of packets. The operation to update T whenever a new packet of x is received is described as follows:

```
function UPDATE FLOW(x)
    if x ∈ T[h1(x)] then                          ▷ x found in bucket h1(x)
        Insert (x, v + 1) in T[h1(x)]             ▷ Update counter and store it again
    else if x ∈ T[h2(x)] then                     ▷ x found in bucket h2(x)
        Insert (x, v + 1) in T[h2(x)]             ▷ Update counter and store it again
    else                                          ▷ x not found in hash table T
        s1 =sizeof(T[h1(x)])                      ▷ Find the size of bucket h1(x)
        s2 =sizeof(T[h2(x)])                      ▷ Fing the size of bucket h2(x)
        if s1 < s2 then                           ▷ Store x into the smallest bucket
            Insert (x, 1) in T[h1(x)]             ▷ s1 is the smallest
        else
            Insert (x, 1) in T[h2(x)]             ▷ s2 is the smallest
        end if
    end if
end function
```

★★★

Exercise 155
Consider a detection system for Distributed Deny of Service (DDoS) attacks that tracks the number of IP packets transferred for each traffic flow. A flow x is identified by the source-destination IP addresses and the port numbers at transport layer (only if present). We assume that the data structure to store the number of packets for each flow is a hash table with 65,536 buckets, specifically designed to exploit the "power of 2 random choices" result for random load balancing.

1. *What is the claim of the "power of 2 random choices" result? Why is it relevant for hash tables?*

2. *Describe in details two hash functions $h_1(x)$ and $h_2(x)$ specifically designed for the flow identifier considered in the problem and for the given size of the hash table.*

3. *Describe in pseudocode the operation to update the hash table when a new packet of flow x is processed.*

★★★

Exercise 156
Consider a traditional hash table with H buckets to store <key,value> elements.

1. *Define the concept of "hash function" and describe its properties.*

2. *Explain the two main relevant results regarding random policies for bins-and-balls models, describing all the involved assumptions.*

3. *Describe two different ways to implement hash tables that exploit the above two results.*

4. *For each of the two implementations:*

   (a) *Describe in pseudocode the insertion of an element; for simplicity, assume that the key does not appear already in the hash.*

   (b) *Describe in pseudocode the lookup of an element.*

   (c) *Evaluate the expected lookup time.*

   (d) *Evaluate the worst case lookup time.*

   (e) *Show an example of insertion of 12 elements when $H = 4$.*

Solution:
Regarding 4a and 4b, let $h(k)$ and $g(k)$ be two hash functions that map a key k into the interval $[1, H]$. Let $T = [T_i]_{i=1}^{H}$ be a table with H buckets, in which T_i is bucket i.

- For traditional hash tables:

```
function INSERT(k, v)   ▷ k is the key and v is the value
    Add (k, v) in T_h(k)                    ▷ Add the element
end function
```

```
function LOOKUP(k)                          ▷ k is the key
    for each (k', v') ∈ T_h(k) do           ▷ Check bucket h(k)
        if k = k' then                      ▷ Key k found
            return v'
        end if
    end for
    return Not-found
end function
```

- For multiple-choice hash tables:

```
function INSERT(k, v)   ▷ k is the key and v is the value
    if |T_h(k)| ≤ |T_g(k)| then     ▷ Find the smallest bucket
        Add (k, v) in T_h(k)                ▷ Add the element
    else
        Add (k, v) in T_g(k)                ▷ Add the element
    end if
end function
```

```
function LOOKUP(k)                                  ▷ k is the key
    for each (k', v') ∈ T_{h(k)} do ▷ Check first bucket h(k)
        if k = k' then                              ▷ Key k found
            return v'
        end if
    end for
    for each (k', v') ∈ T_{g(k)} do   ▷ Check second bucket g(k)
        if k = k' then                              ▷ Key k found
            return v'
        end if
    end for
    return Not-found
end function
```

★★★

Exercise 157
Consider a traditional hash table of size t to store s <key,value> elements.

1. *Define the concept of "hash function" and describe its properties.*
2. *Describe how an insertion is performed.*
3. *Describe how a lookup is performed.*
4. *Evaluate the expected lookup time.*
5. *Evaluate the worst case lookup time.*
6. *Show an example of insertion of 10 elements when $t = 5$.*
7. *Is it possible to devise an enhanced version of hash table with better worst case lookup time? If yes, how?*

★★★

Exercise 158
Consider the adoption of the fingerprinting scheme in data storage system.

1. *Describe how fingerprinting works.*
2. *Describe the problem of false positives.*
3. *Describe how deletion is supported.*
4. *Compute analytically the minimum number of bit per fingerprint to achieve a probability of false positive equal to ϵ. Define* every *notation adopted in the proof and explain all the steps.*
5. *Assume to store 1000 elements in a bit array with fingerprinting. What is the size of the storage to achieve a probability of false positive equal to 10^{-6}?*

★★★

Solution:
5. As a result of the formula achieved in question 4, the total number of bits per fingerprint is $b = \log_2(m/\epsilon)$ where $m = 10^3$ and $\epsilon = 10^{-6}$, thus $b = \log_2(10^9) = 3\log_2(1000) \approx 30$ bits per fingerprint. Thus, the total memory size for the bit array is 10^9 bits, equivalent to 125 MB.

Exercise 159

Consider an Internet traffic monitoring system based on a b-bit fingerprint $F(x)$ for flow x, adopting a bit string hash for the storage. The flow is identified by the couple $x = (IP_{source}, IP_{destination})$. Answer to the following question:

1. *What is the total amount of memory (in kbytes) required for storage in the case $b = 32$ after $1,000$ flows have been inserted?*
2. *Describe a possible fingerprint function when $b = 32$*
3. *Prove the formula to compute the probability of false positive in function of b and of the number k of inserted flows*
4. *What would be the minimum value of b to guarantee a probability of false positive less than ϵ?*
5. *Is there any case in which the bit string array shows a null probability of false positive?*
6. *Show the pseudocode to insert a generic flow x into the monitoring system, after defining properly all the involved data structures.*
7. *Show the pseudocode to search a generic flow x*
8. *Show the pseudocode to delete a generic flow x*
9. *Does the considered monitoring system supports correctly the deletion operation? Why?*

Solution:

1. The bit string hash requires 2^b bits, independently from the stored elements. In the considered case, around 4 Gbit ($\approx$ 500 Mbytes) are required.
2. A possible hash function can be obtained by the EXOR between the source and destination IP address: $F(x) = \text{IP}_{\text{source}} \oplus \text{IP}_{\text{destination}}$.
3. The probability that a specific bit in the bit array equals to one when adding a flow is $p = 1/2^b$. Consider now a flow y which has not been inserted before. Now the probability that $F(y) \neq F(x)$ for any x already inserted is $(1-p)^k$. Thus, the probability of false positive is
$$\Pr(\text{false positive}) = 1 - \left(1 - \frac{1}{2^b}\right)^k \approx 1 - e^{-\frac{k}{2^b}}$$
4. By setting $\Pr(\text{false positive}) < \epsilon$, after some simple steps, we get
$$b > \log_2 \frac{k}{-\log_2(1-\epsilon)}$$
5. Only when no element has been inserted in the bit array, the probability of false positive is null.
6. Let BSA be the string array with 2^b bits, and $BSA[i] \in \{0,1\}$ be the ith bit in the data structure. Let $F(x)$ be the fingerprint of flow x.

```
function INSERT(x)
    BSA[F(x)] = 1
end function
```

7.
```
function SEARCH(x)
    if BSA[F(x)] = 1 then
        return FOUND
    else
        return NOT_FOUND
    end if
end function
```

8.
```
function DELETE(x)
    BSA[F(x)] = 0
end function
```

9. If the deletion operation occurs, then the bit string array can introduce false negatives and thus the deletion *must not be supported*.

★★★

Exercise 160

Consider an intrusion detection system that stores all the source IP addresses observed at the network interface of a router, using a table implemented with a simple vector array. A fingerprinting scheme is adopted. Let $x = a.b.c.d$ *be an IP address expressed with the canonical representation with four decimal digits; the corresponding fingerprint is* $F(x) = (a + b + c + d)$ *modulus* 16.

1. *Consider the sequence of operations in the above table. For each operation, show the content of table* T *and the probability of false positive after the operation. Assume the table initially empty:* $T = \{\}$.

Time	*Operation*	*Table* T	*Prob. false positive*
1	*insert(1.2.3.4)*		
2	*insert(2.3.4.5)*		
3	*insert(2.3.4.1)*		
4	*insert(3.4.5.2)*		
5	*delete(1.2.3.4)*		
6	*delete(2.3.4.1)*		
7	*delete(3.4.5.2)*		
8	*delete(2.3.4.5)*		

2. *In the above sequence of operations, add at some time an operation for which a false positive event occurs.*

3. *Assume that* k *IP addresses are stored exploiting fingerprints of* b *bits. What is the total required storage in terms of bits? What is the final probability of false positives?*

4. *Assume that* k *IP addresses are stored directly in the table without fingerprinting. Compare this solution with the one adopting fingerprinting in terms of storage requirement and probability of false positives.*

5. *Show the pseudo-code to insert, delete and search an element in a table exploiting fingerprinting.*

Solution:

1. The evolution of the table is reported below. Note that multiple copies of the same fingerprint must be stored to support deletion.

Time	Operation	Table T	Prob. false positive
1	insert(1.2.3.4)	$\{10\}$	1/16
2	insert(2.3.4.5)	$\{10, 14\}$	1/8
3	insert(2.3.4.1)	$\{10, 10, 14\}$	1/8
4	insert(3.4.5.2)	$\{10, 10, 14, 14\}$	1/8
5	delete(1.2.3.4)	$\{10, 14, 14\}$	1/8
6	delete(2.3.4.1)	$\{14, 14\}$	1/16
7	delete(3.4.5.2)	$\{14\}$	1/16
8	delete(2.3.4.5)	$\{\}$	0

2. E.g., Search(2.1.4.3) at time 1.5 or Search(1.2.3.4) at time 5.5 would lead to a false positive event.

3. The total storage is bk bits.The final probability of false positives is

$$1 - \left(1 - \frac{1}{2^b}\right)^k$$

4. Without fingerprinting, each IP address is represented with 32 bits. Thus the storage requirement is $32k$ bits in total. Note that this data structure avoids false positive events, but requires a larger storage which is $32/k$-times the one with fingerprinting (assuming $b < 32$).

5. Assume that the size of the vector array T is n. Assume that all the entries in T have been initialized to -1 as default value.

```
function INSERT(x)
    for i = 1 → n do
        if T[i] = −1 then
            T[i] = F(x)
            return OK
        end if
    end for
    return ERROR_FULL_TABLE
end function
```

```
function FIND-ELEMENT(x)
    for i = 1 → n do
        if T[i] = F(x) then
            return FOUND
        end if
    end for
    return NOT_FOUND
end function
```

```
function DELETE-ELEMENT(x)
    for i = 1 → n do
        if T[i] = F(x) then
            T[i] = −1
            return OK
        end if
    end for
    return ERROR_NOT_FOUND
end function
```

★★★

Exercise 161

Consider a flow-monitor tool that stores all the active flows using a table implemented with a simple vector array. A fingerprint scheme is adopted to map any flow to an integer in the interval $[0, 1023]$; the fingerprint is designed to identify flows independently from their direction, i.e. the two flows of a bidirectional communication are identified by the same value. Let a flow f be defined in terms of a 4-uple with the notation reported in the following table:

Source IP address	$x_1.x_2.x_3.x_4$
Destination IP address	$y_1.y_2.y_3.y_4$
Layer-4 source port	p_x
Layer-4 destination port	p_y

1. *Design a fingerprint scheme $F(f)$ that satisfies all the design constraints above and able to deal also with flows without any layer-4 port.*
2. *Consider the sequence of operations in the above table. For each operation, show the content of table T and the probability of false positive after the operation. Assume the table initially empty: $T = \{\}$.*

Time	*Operation*	*Table T*	*Prob. false positive*
1	*insert(1.2.3.4, 5.6.7.8, 1, 2)*		
2	*insert(2.3.4.5, 6.7.8.9, 3, 4)*		
3	*insert(2.3.4.5, 9.8.7.6, 5, 2)*		
4	*insert(3.4.0.3, 5.7.9.5, 2, 1)*		
5	*delete(1.2.3.4, 5.6.7.8, 1, 2)*		
6	*delete(2.3.4.5, 9.8.7.6, 5, 2)*		
7	*delete(3.4.0.3, 5.7.9.5, 2, 1)*		
8	*delete(2.3.4.5, 6.7.8.9, 3, 4)*		

3. *In the above sequence of operations, add an operation at some specific time $t \in [0, 10]$ for which a false positive event occurs. Note that the false positive can occur for other values of t. What is the largest time interval during which the false positive would occur?*
4. *Show the pseudo-code to insert, delete and search an element in a table exploiting fingerprinting.*

Solution:

1. The fingerprint can be defined as follows:

$$F(f) = (x_1 + x_2 + x_3 + x_4 + y_1 + y_2 + y_3 + y_4 + p_x + p_y) \mod 1024$$

   where $p_x = p_y = 65,536$ whenever layer-4 port is not defined.

2. Thus the evolution of the hash table is the following:

Time	Operation	Table T	Prob. false positive
1	insert(1.2.3.4, 5.6.7.8, 1, 2)	$\{39\}$	$1/1024$
2	insert(2.3.4.5, 6.7.8.9, 3, 4)	$\{39, 51\}$	$2/1024$
3	insert(2.3.4.5, 9.8.7.6, 5, 2)	$\{39, 51, 51\}$	$2/1024$
4	insert(3.4.0.3, 5.7.9.5, 2, 1)	$\{39, 39, 51, 51\}$	$2/1024$
5	delete(1.2.3.4, 5.6.7.8, 1, 2)	$\{39, 51, 51\}$	$2/1024$
6	delete(2.3.4.5, 9.8.7.6, 5, 2)	$\{39, 51\}$	$2/1024$
7	delete(3.4.0.3, 5.7.9.5, 2, 1)	$\{51\}$	$1/1024$
8	delete(2.3.4.5, 6.7.8.9, 3, 4)	$\{\}$	0

3. E.g., Search(2.3.0.9, 10.7.0.1, 10, 9) during any time $t \in (2, 8)$ would lead to a false positive.

4. See the solution of point 5 of Ex. 160.

5.2 Bloom filters

Exercise 162
Consider a Bloom filter of 10 *bits and* 3 *hash functions.*

1. *Which kind of data structure does the Bloom filter implement?*
2. *Describe some possible scenarios in networks where bloom filters are adopted.*
3. *Write in pseudocode the operations of "write" and "read" in a bloom filter.*
4. *Show an example of "false positive" event.*

★★★

Exercise 163
Consider a Bloom filter with 10^6 *bits available for storage and* 3 *hash functions.*

1. *Describe which data structure a Bloom filter implements.*
2. *Describe some possible scenarios in networks where bloom filters are adopted.*
3. *Write in the pseudocode the operations of "write" and "search" in such bloom filter.*
4. *Define the probability of false positive.*
5. *Write the formula of the probability of false positive for the specific Bloom filter above.*

★★★

Exercise 164
Consider a Bloom filter with 800 *kbits available for storage and* 4 *hash functions.*

1. *Describe which data structure a Bloom filter implements.*
2. *Describe some possible scenarios in networks where bloom filters are adopted.*
3. *Write in the pseudocode the operations of "write" and "search" in such bloom filter.*
4. *Define the probability of false positive.*
5. *Consider the probability of false positive shown in Fig. 5.1. After having defined* k, n *and* m*, explain the meaning of the graph applied to the specific Bloom filter considered in the question.*

Solution:
For the first four questions, refer to the class notes.
Regarding question 5, let k be the number of hash functions, n be the memory size and m be the number of elements already stored in the Bloom filter.
In this specific filter considered in the question, $k = 4$ and $n = 800$ kbit. Thus, Fig. 5.1 shows that

- after inserting $m = 1\,10^5$ elements (i.e., $n = 8m$), the probability of false positive is 0.023;

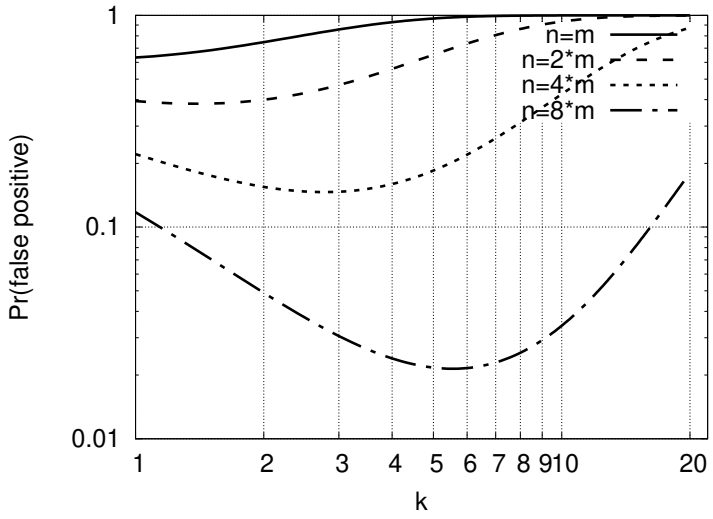


Figure 5.1: Probability of false positive (Ex. 164)

.

- after inserting $m = 2\,10^5$ elements (i.e., $n = 4m$), the probability of false positive is 0.16;
- after inserting $m = 4\,10^5$ elements (i.e., $n = 2m$), the probability of false positive is 0.56;
- after inserting $m = 8\,10^5$ elements (i.e., $n = m$), the probability of false positive is 0.93.

Thus, after storing 100 thousands elements in the Bloom filter, the probability of false positive starts to become relatively large (i.e., approximatively larger than 0.1).

★★★

Exercise 165
Consider a network monitoring application that stores the connection identifiers of all UDP and TCP flows that have been observed in a particular network probe. Assume that the maximum number of flows to track is 1024.

1. *Define the exact and the approximated set membership problem for this particular problem.*
2. *Discuss a data storage solution that solves the exact problem.*
3. *Discuss a data storage solution that solves the approximated problem with bit string hashing.*
4. *Compare the two solutions.*

★★★

Exercise 166
Consider a YouTube monitoring system, based on a memory of 10 *kbytes, which stores the IP addresses of the YouTube users. At a particular time, around* $4,000$ *flows have been stored. Assume that the memory implements either (i) a Bloom filter, (ii) a bit string hash.*

In the case of a Bloom filter:

1. *How does the filter work?*

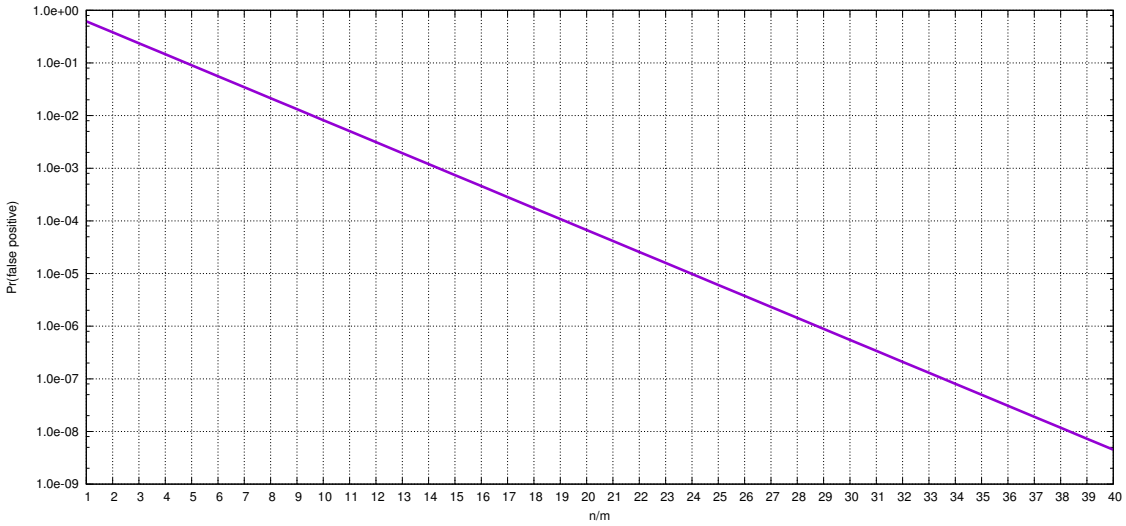


Figure 5.2: Probability of false positive in a bloom filter

2. *Write in pseudocode the operations of "add" and "search" in such bloom filter.*
3. *What is the probability of false positive? Consider the performance shown in Fig. 5.2.*

In the case of a bit string hash:

1. *How does the hash work?*
2. *Show analytically what is the minimum number of bits per fingerprint to achieve a very small false positive probability.*
3. *What is the probability of false positive? Is it better or worse than the one achieved by the Bloom filter?*

Solution:

The bloom filter uses k hash functions to store a given element x in a bit string of size n. Assume that BF is a bit array initialized to 0: $BF(i) = 0$ for all $i = 1, \ldots, n$. Assume that we have k hash functions $h_1(x), \ldots, h_k(x)$.

To add an element x:

```
function ADD(x)
    for i = 1 → k do
        BF(h_i(x)) ← 1
    end for
end function
```

To check whether an element x is present:

```
function SEARCH(x)
    for i = 1 → k do
        if BF(h_i(x)) == 0 then
            return false
        end if
    end for
    return true
end function
```

In the considered scenario, the size of the bit array is $n = 80,000$ bits whereas the number of elements is $m = 4,000$. Assuming the optimal number of hash functions, from the graph the probability of false positive is less than 10^{-4}.

In the second case, the bit string hash uses just one hash function to store the element into a fingerprint table. When adding a new element x, the corresponding fingerprint $h(x)$ is stored in the table. To check whether an element x is present, the corresponding $h(x)$ is searched within the table. To optimize such procedure, usually a binary search or an hash table is adopted.

Assume that b is the number of bit for each fingerprint, i.e. $h(x) \in [0, \ldots, 2^b - 1]$. It is possible to show that b must be at least $\Omega(\log m)$ to get a probability of false positive $< \epsilon$. When $b = 2 \log_2 m$,

$$\Pr(\text{false positive}) = 1 - \left(1 - \frac{1}{m^2}\right)^m \approx 1 - (e^{-1/m^2})^m = 1 - e^{-1/m}$$

which can be approximated by $1/m$. In the considered scenario, the probability of false positive is $1/4000 = 2.5\ 10^{-4}$ which is larger than the case obtained with a Bloom filter.

★★★

Exercise 167
Assume a Bloom filter with 10 bits used to store integer numbers. The filter uses the following three hash functions: $h_1(x) = x \mod 10$ *,* $h_2(x) = (x + 4) \mod 10$*,* $h_3(x) =$ *floor*$(\log_{10} x)$*, where* $\mod$ *is the modulo operator.*

1. *Are these 3 hash functions "good" for the considered Bloom filter? Why?*
2. *Add the following values, showing the state of the Bloom filter after each insertion: 10001, 2007, 35.*
3. *Which operation (if any) would lead to a false positive event for the above Bloom filter, after the 3 insertions?*
4. *Which operation (if any) would lead to a false negative event for the above Bloom filter, after the 3 insertions?*
5. *Explain with an example why the deletion is not allowed in the above Bloom filter, after the 3 insertions.*

Solution:

1. $h_1(x)$ and $h_2(x)$ are not independent, so the two hash functions perform poorly for a Bloom filter
2. after inserting 10001, $BF = [0100110000]$; after inserting 2007, $BF = [0101110100]$; after inserting 35, $BF = [0101110101]$.
3. search(11) will lead to a false positive
4. delete(35) and then search(10001) would lead to a false negative, thus deletion is not allowed
5. see above

★★★

Exercise 168
Consider a caching scheme that adopts a Bloom filter to track the content stored in a cache. The bloom filter is implemented in a fast memory of 100 *kbytes and must ensure a probability of false positive less than* 10^{-4}*. Consider the performance shown in Fig. 5.2.*

1. *How does the filter work?*
2. *Write in pseudocode the operations of "add" and "search" in such bloom filter.*
3. *What are the consequences of a false positive event in the considered caching scheme?*
4. *What is the maximum number of contents that can be inserted in the bloom filter (i.e., stored in the cache), without violating the probability of false positive?*

Solution: The problem is almost identical to Ex. 166.

To ensure the probability of false positive less than 10^{-4}, it must be $n/m \geq 20$. In this scenario, $n = 8\ 10^5$ bit, thus $m \leq n/20 = 40,000$, i.e. the maximum number of contents that can be stored is $40,000$.

★★★

Exercise 169
Consider a network security system, based on a memory of 10 Mbytes, which stores a white list of IP addresses that are allowed to communicate in the network. Assume that the memory implements a Bloom filter, designed to keep the probability of false positive less than 10^{-6}. Consider the performance shown in Fig. 5.2.

1. *How does the filter work?*
2. *Write in pseudocode the operations of "add" and "search" in such bloom filter.*
3. *What are the consequences of a false positive event in the considered network security system?*
4. *What is the maximum number of IP addresses that can be stored in the whitelist, without violating the probability of false positive?*

Solution:

The problem is almost identical to Ex. 166.

From the graph, $n/m = 29$ to ensure the requested false positive probability. Since the storage is $n = 80$ Mbits, then the maximum number of IP addresses that can be stored is $80\ 10^6/29 \approx 2.7\ 10^6$.

5.3 Cuckoo filters

Exercise 170 (out of scope for the program 2021/22)
Consider a Cuckoo filter.

1. *What is the purpose of it?*
2. *Describe some application scenario.*
3. *Explain the main ideas exploited to design it.*
4. *What are the properties of the adopted hash function/s?*
5. *Show an example of insertion of 6 elements into a Cuckoo filter based on a table with 4 rows and 2 columns.*
6. *Describe how the deletion works.*
7. *Describe the advantages and disadvantages of Cuckoo filters with respect to Bloom filters.*

★★★

Exercise 171 (out of scope for the program 2021/22)

1. *Consider a Cuckoo hash:*
   (a) *What is its purpose?*
   (b) *How does it work?*
   (c) *Show an example of element relocation.*
2. *Consider a Cuckoo filter:*
   (a) *What is its purpose?*
   (b) *How does it work?*
   (c) *What is the difference between a Cuckoo hash and a Cuckoo filter?*