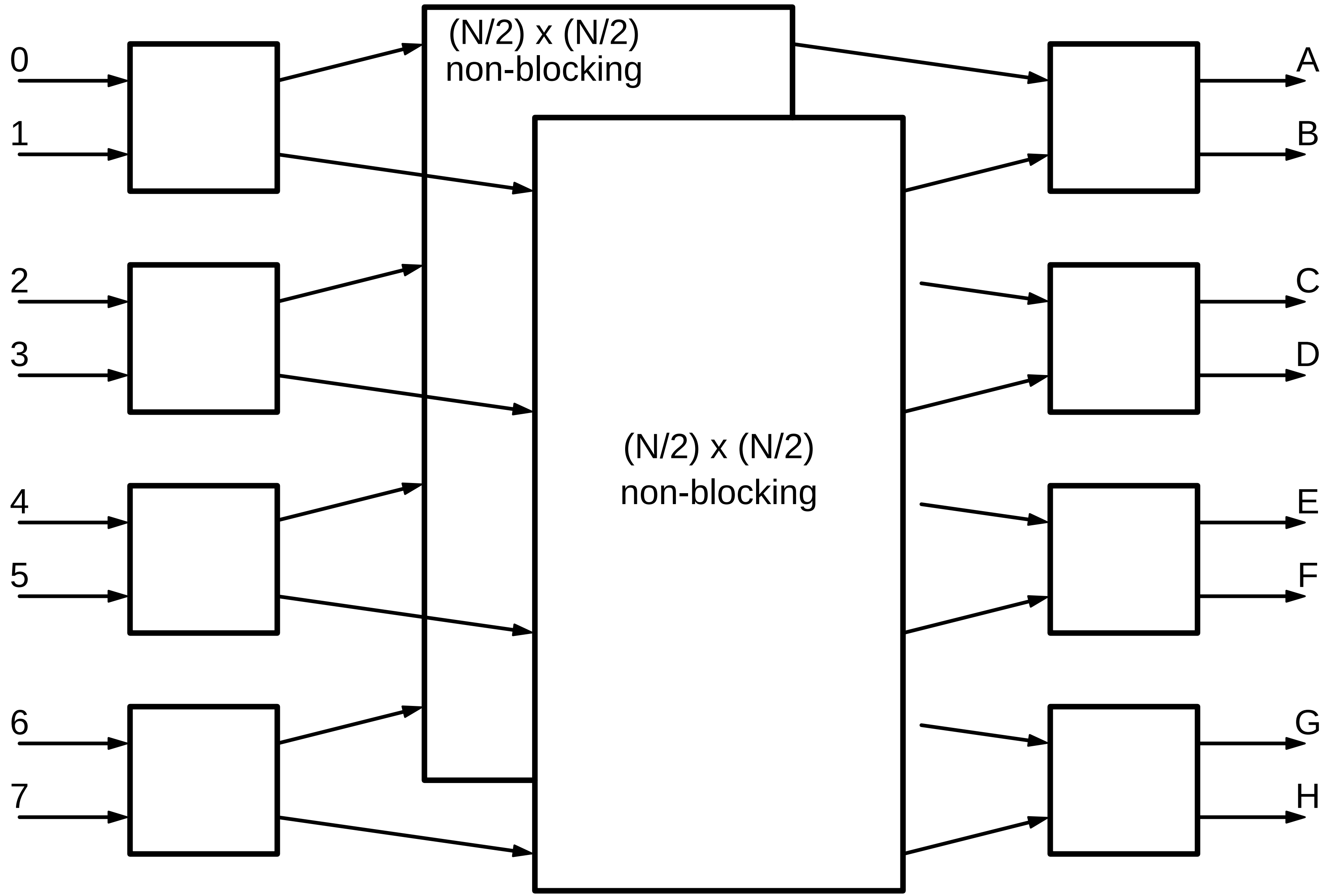
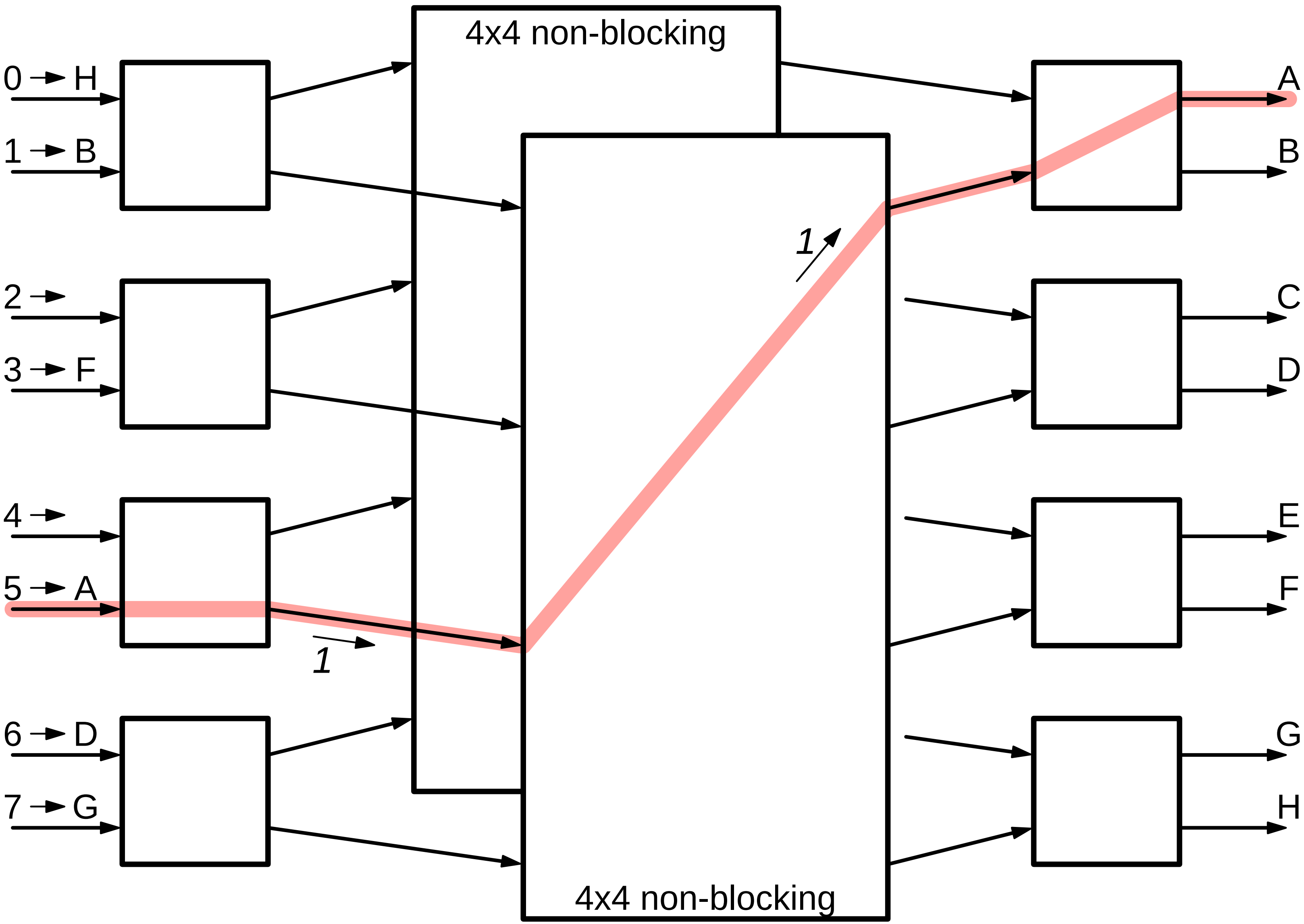
# **5.2** Switching Fabric Topologies

- Benes Fabrics – recursive construction
  - rearrangeably non-blocking (probably the lowest cost such)
  - proof under circuit switching with 2×2 switches
- Banyan Fabrics – one half of a Benes
  - internal blocking for non-uniform traffic patterns
- Clos Networks – generalization of Benes
  - rearrangeably or strictly non-blocking, depending on midle-stage width
- Fat Trees – like folded, bidirectional Clos
- Others
  - meshes, tori, hypercubes, etc.

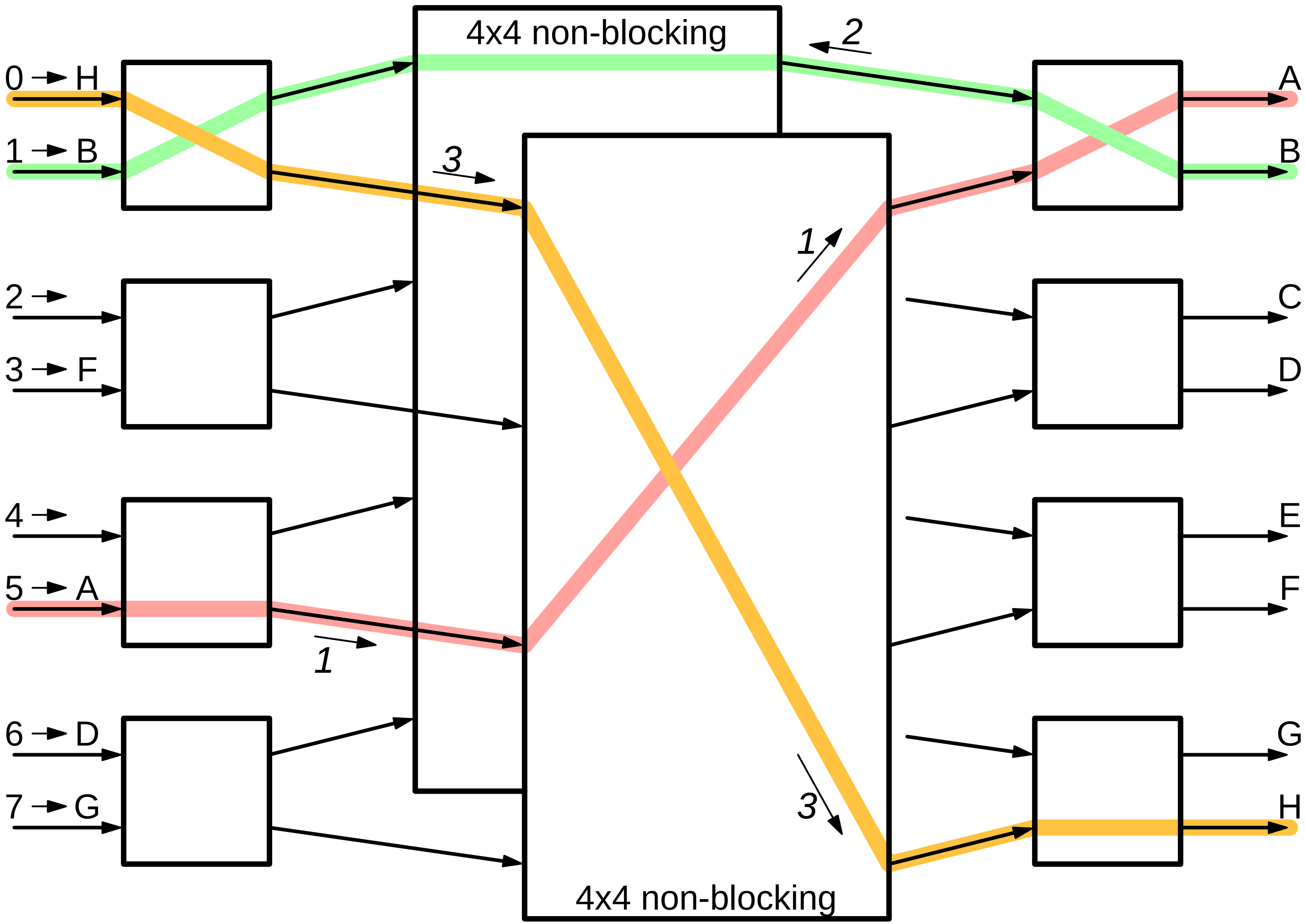
## *Benes Network: Recursive Definition*



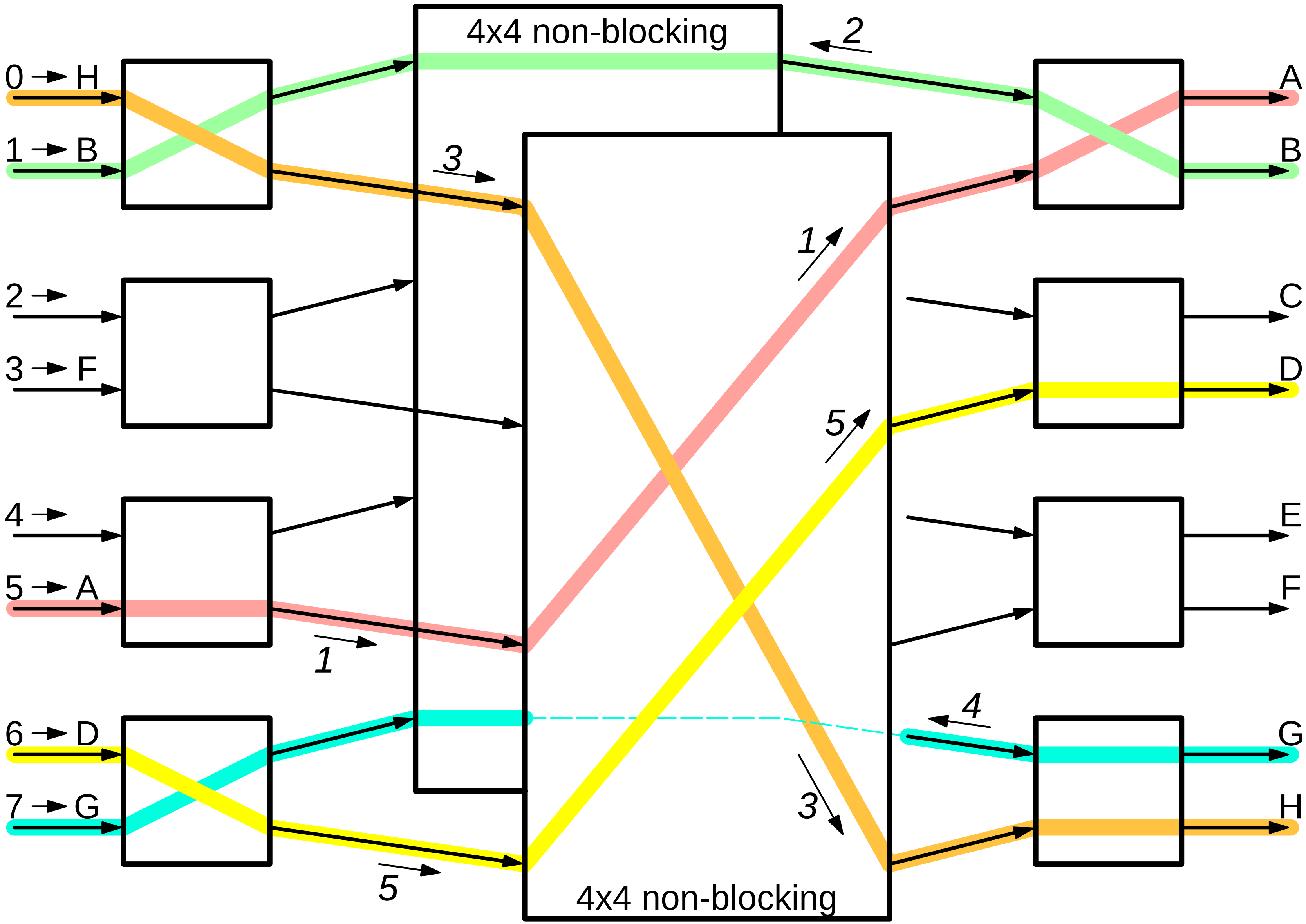

NxN Benes network: rearrangeably non-blocking



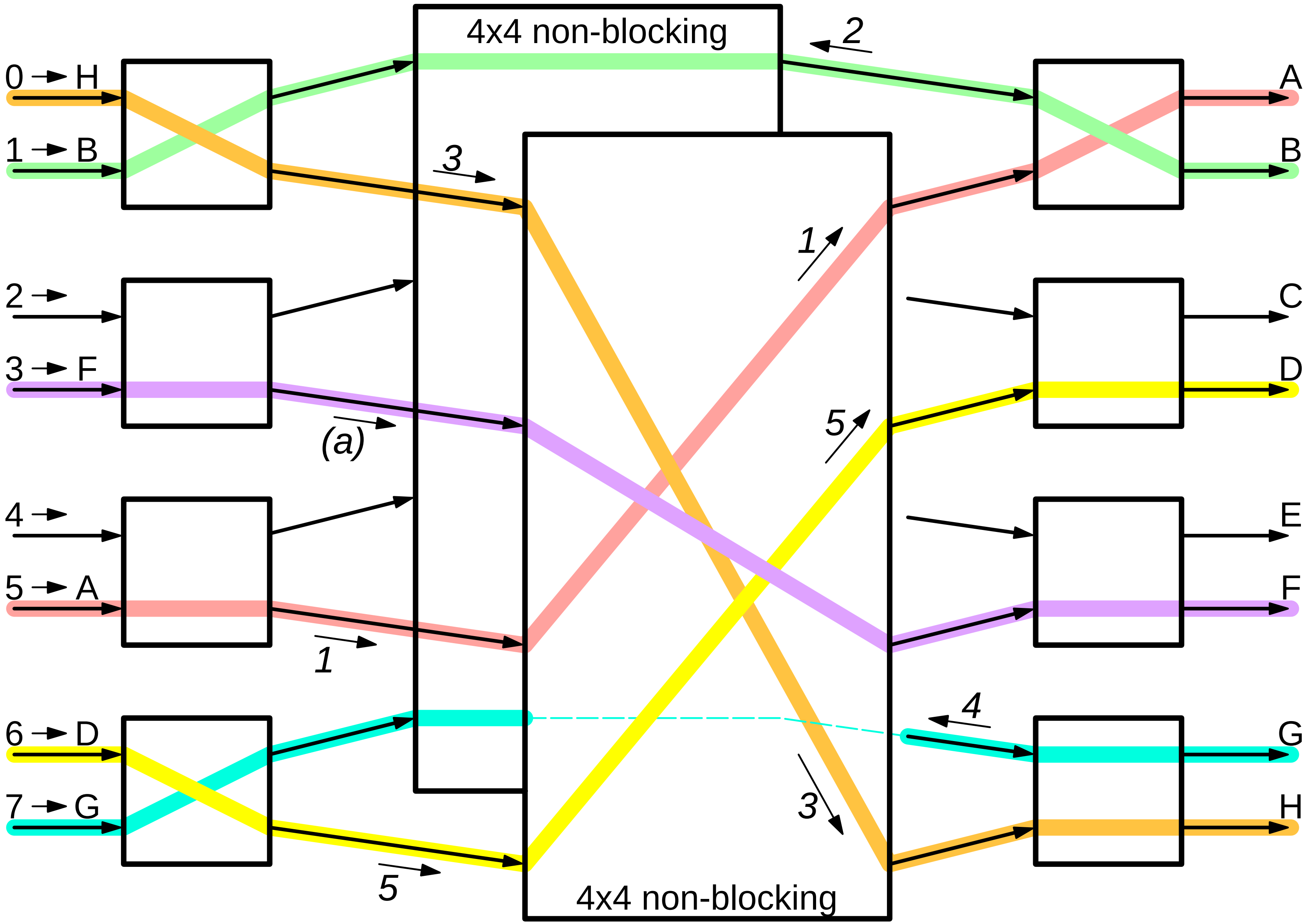

- Circuit Connections: Start from an input, use one of the subnets




- Continue from the brother port of the output, then the brother of the input

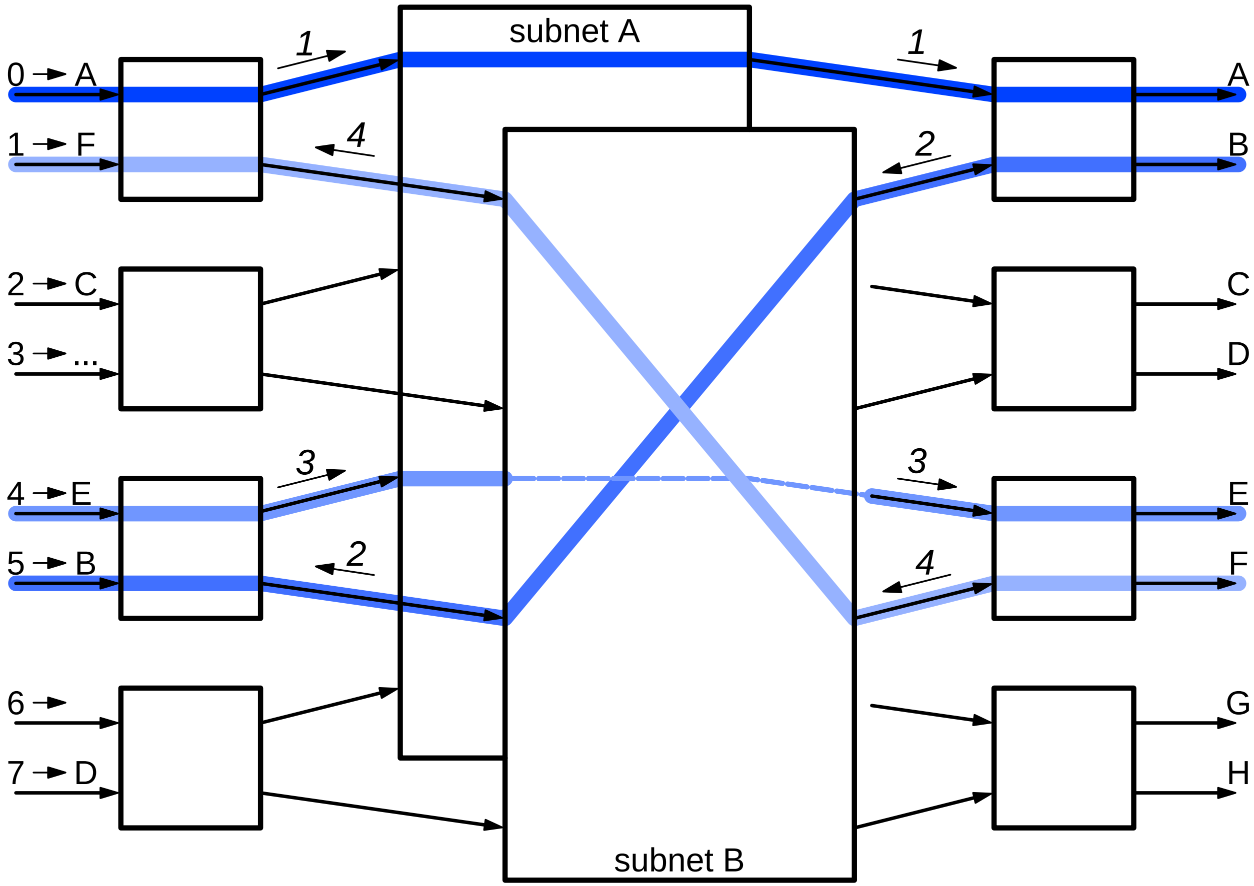



- Keep “threading” output and input switches, till closing or no-connection




- Start a new “thread” (a) from an unconnected input, till completing all conn.

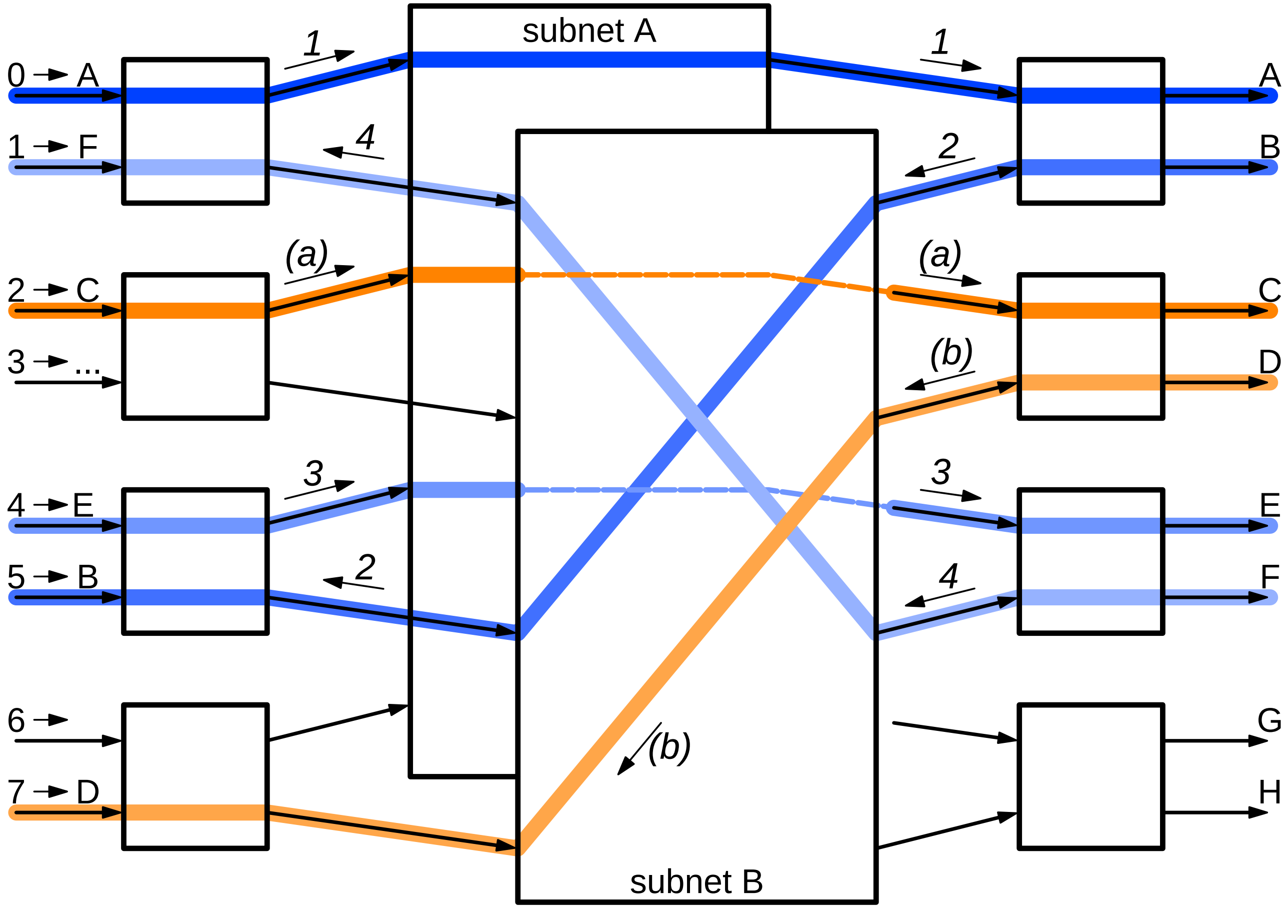
# (A) Thread termination on input side (1 of 2)




- Threads always start on the input side
- If a thread terminates on the input side:
  - all touched output switches are completely connected
  - concerning touched input switches:

    (1) if thread closes, all are complete, …

# (A) Thread termination on input side (2 of 2)

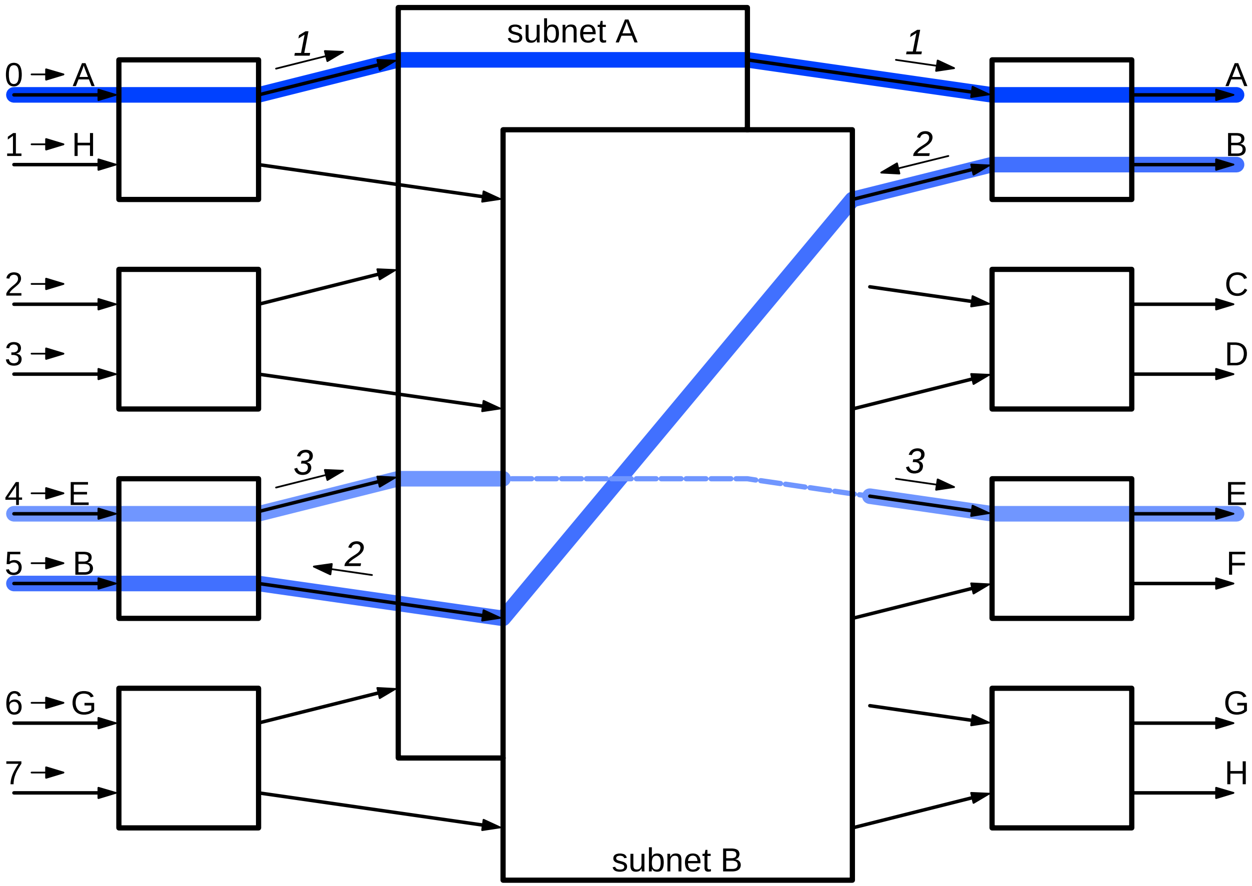



- Threads always start on the input side
- If a thread terminates on the input side:
  - –all touched output switches are completely connected
  - –concerning touched input switches:

    (1) if thread closes (4), all are complete,

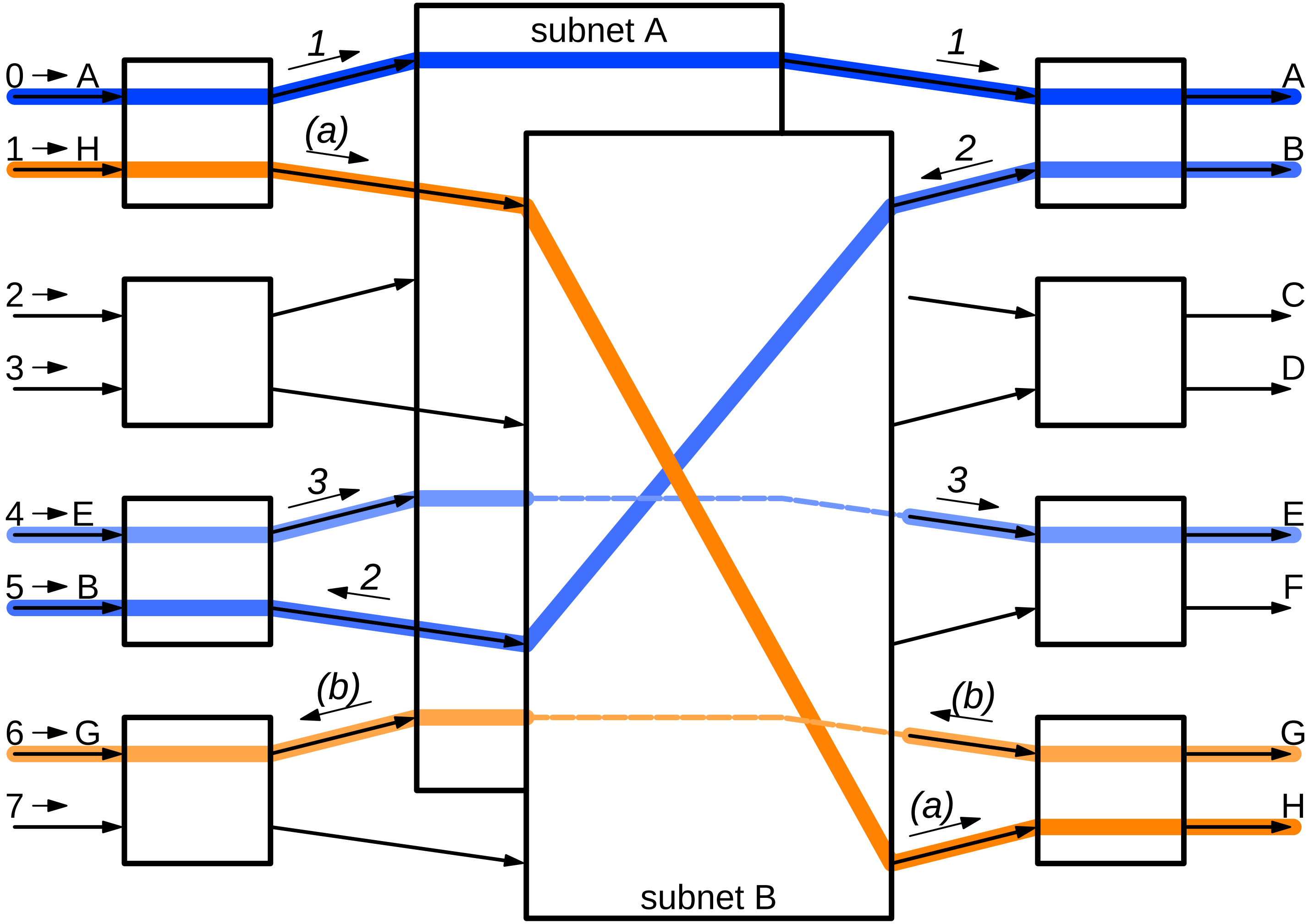
    (2) if thread terminates on half-used input (b): all touched input switches are complete, except the first one, which is half-covered by this thread

# (B) Thread termination on output side




- Threads always start on the input side
- If a thread terminates on the output side:
  - all touched output switches are completely connected
  - the first touched input switch is half-covered

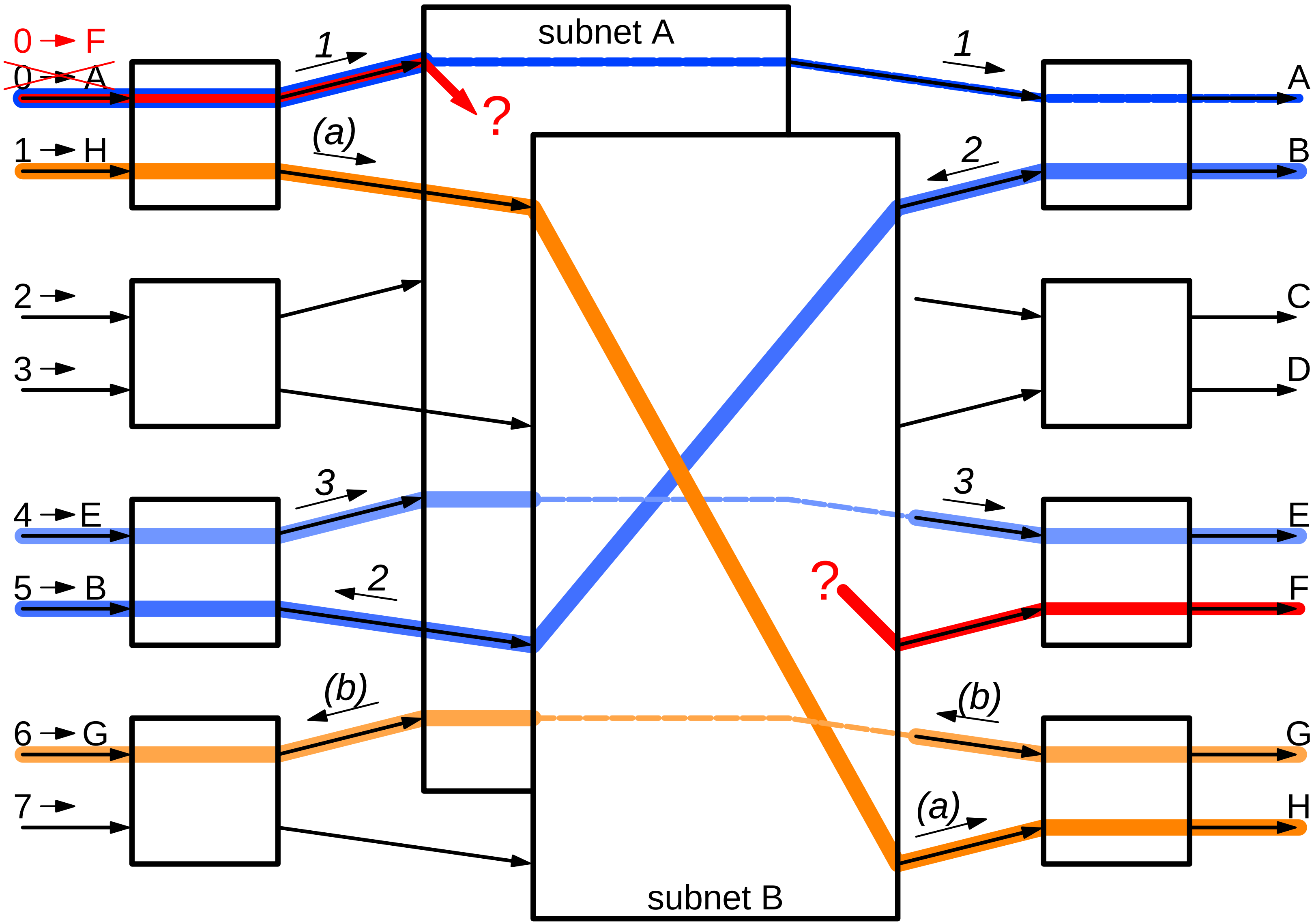
# (C) Completing half-covered input switches



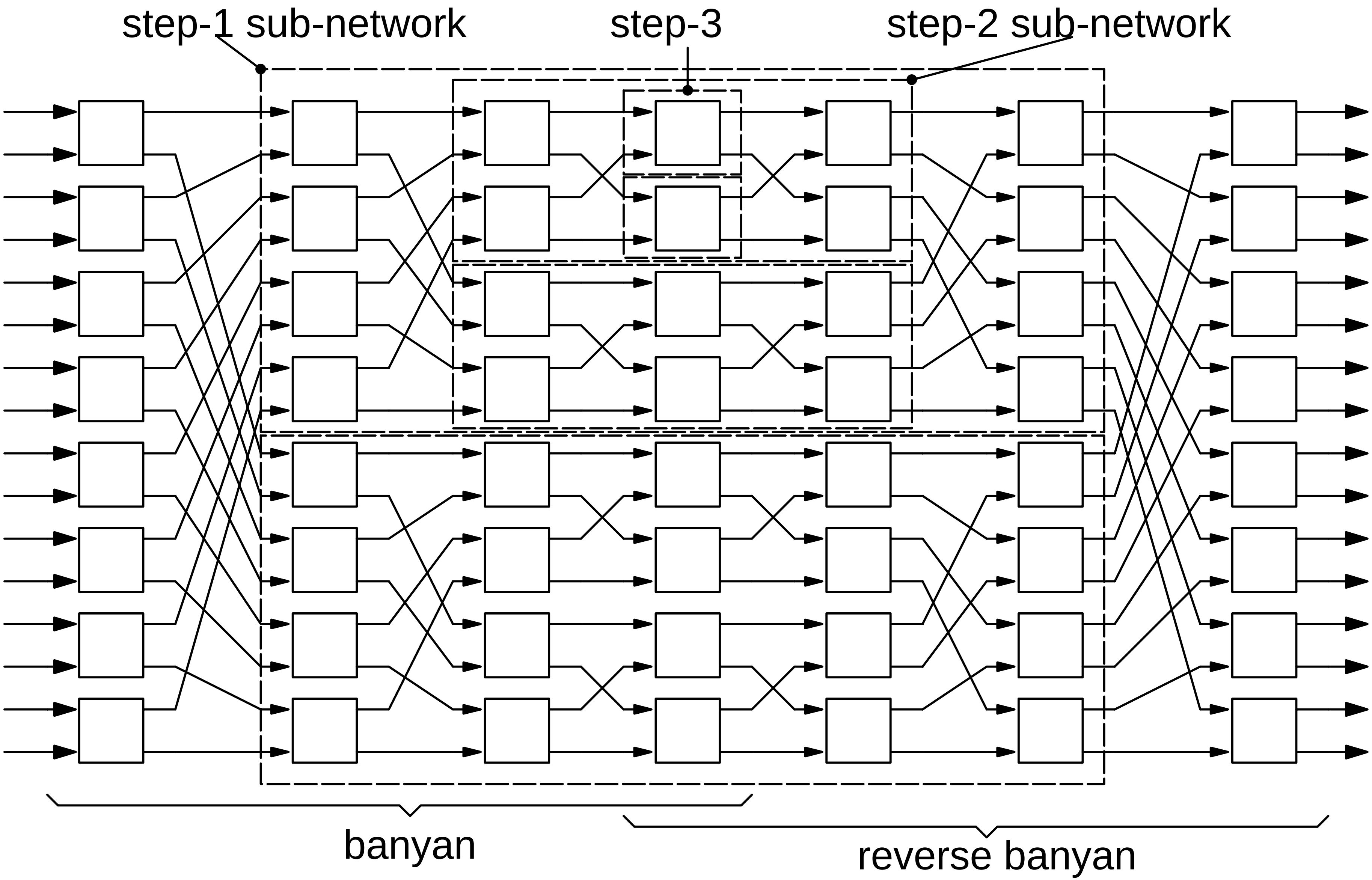

- New threads always start from a half-covered input switch, if there is one

  ⇒ all threads cover all out-sw's they touch, in-sw's are covered in sequence

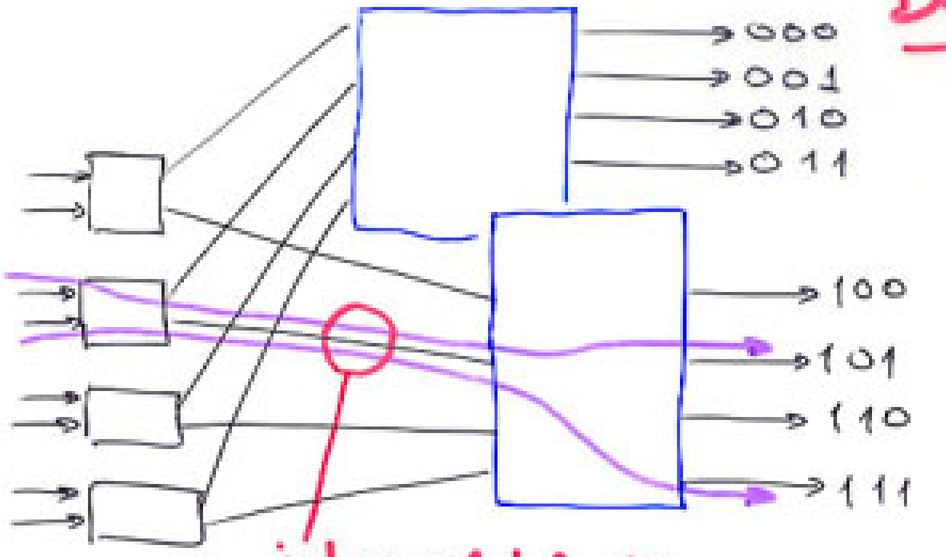
# Benes Fabric: *Rearrangeably* Non-Blocking

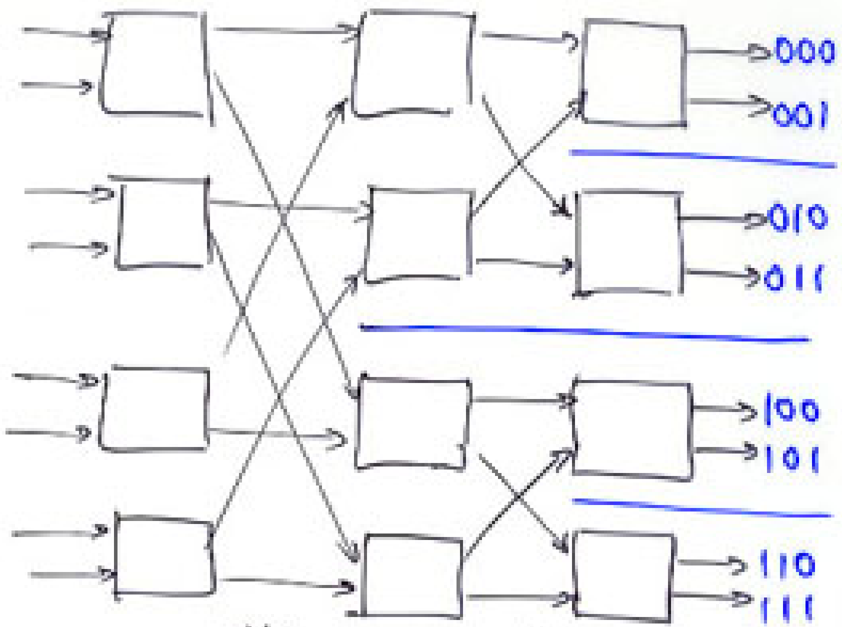


# *Recursive Construction of 16x16 Benes Network out of 2x2 Switches*



# Banyan (or Butterfly) Network




- single path from input to output
- routing decision based on single bit of destination address – each stage looks at a different bit.
- NxN network has $\log_2 N$ stages of $\frac{N}{2}$ switches (2x2) each
- Number of states of the network =

$$= 2^{(\#\text{ of binary switches})} = 2^{(\log N)\cdot\left(\frac{N}{2}\right)} = \left(2^{\log N}\right)^{\left(\frac{N}{2}\right)}$$

$$= N^{\left(\frac{N}{2}\right)} = \left(\frac{N}{2}\cdot 2\right)^{\frac{N}{2}} =$$

$$= \underbrace{\frac{N}{2}\cdot\frac{N}{2}\cdot\frac{N}{2}\cdots\frac{N}{2}}_{\frac{N}{2}}\cdot\underbrace{2\cdot 2\cdot\ldots\cdot 2}_{\frac{N}{2}} <$$

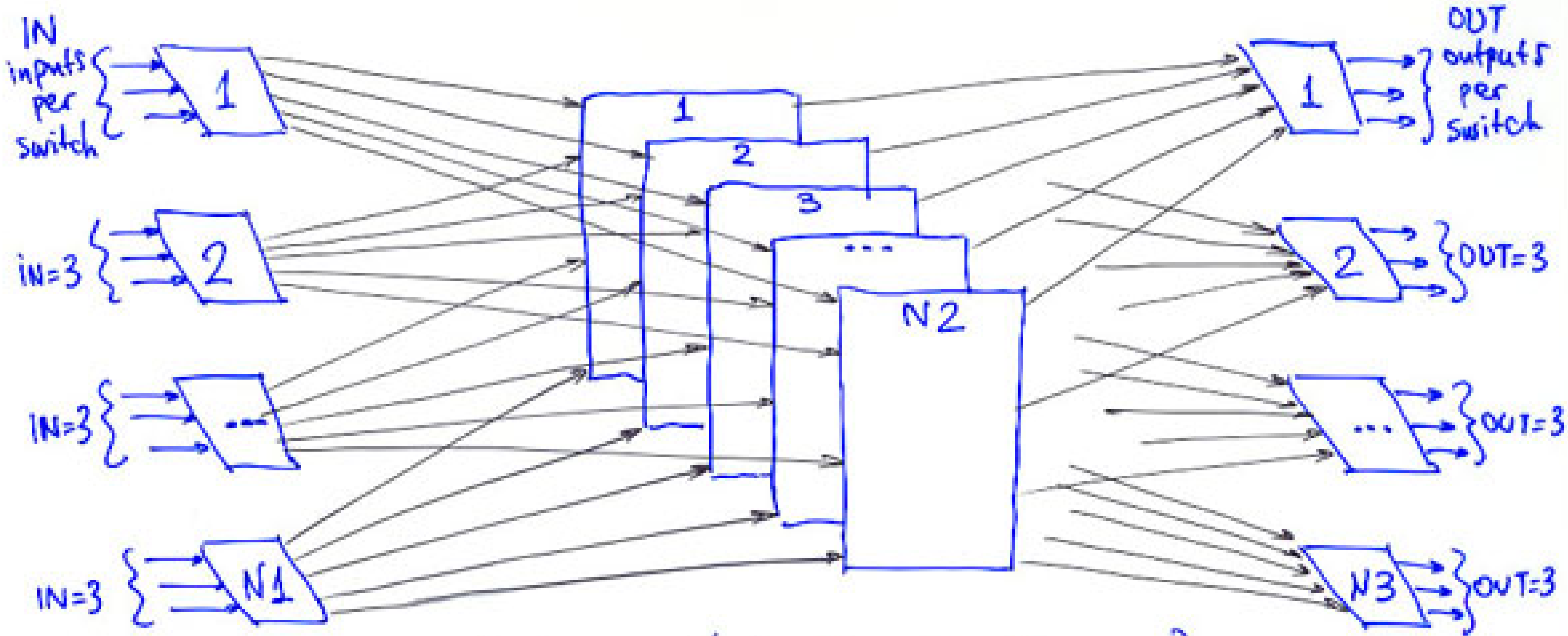
$$< N\cdot\underbrace{(N-1)\cdot(N-2)\cdots\left(\frac{N}{2}\right)}_{\frac{N}{2}}\cdot\underbrace{\left(\frac{N}{2}-1\right)\cdots 2\cdot 1}_{\frac{N}{2}} = N!$$

⇒ #States < N! ⇒ cannot route all permutations!

# Reminder of topics discussed in class, which need to be included in future-year slides

- Banyan fabric is like a set of binary trees with shared nodes.
- Dst/Src Address: in a banyan, a single field suffices for both, if each stage replaces one "consumed" destination-address bit with one corresponding source-address bit.
- Number of states of a fabric made of 2×2 switch elements, internal blocking, and cost-minimality of non-blocking fabrics:
  – banyan has $(½ \cdot N \cdot log_2N)$ switches, and its number of states is less than $(N!)$, hence it cannot route all permutations – has internal blocking
  – Benes has $(N \cdot (log_2N - ½))$ switches, its number of states is more than $(N!)$, it can route all permutations, and it is internally non-blocking
  – the minimum-cost non-blocking fabric made of 2×2 switches must have at least $(log_2(N!))$ switches, hence it must have a cost in-between the cost of the banyan and the cost of the Benes.
- Clos Networks: *N2* parameter is like "internal speedup"…
- Fat Trees – Benes/Clos: bidirectional vs. unidirectional links:
  – bidirectional links needed to carry flow control (backpressure) info. …

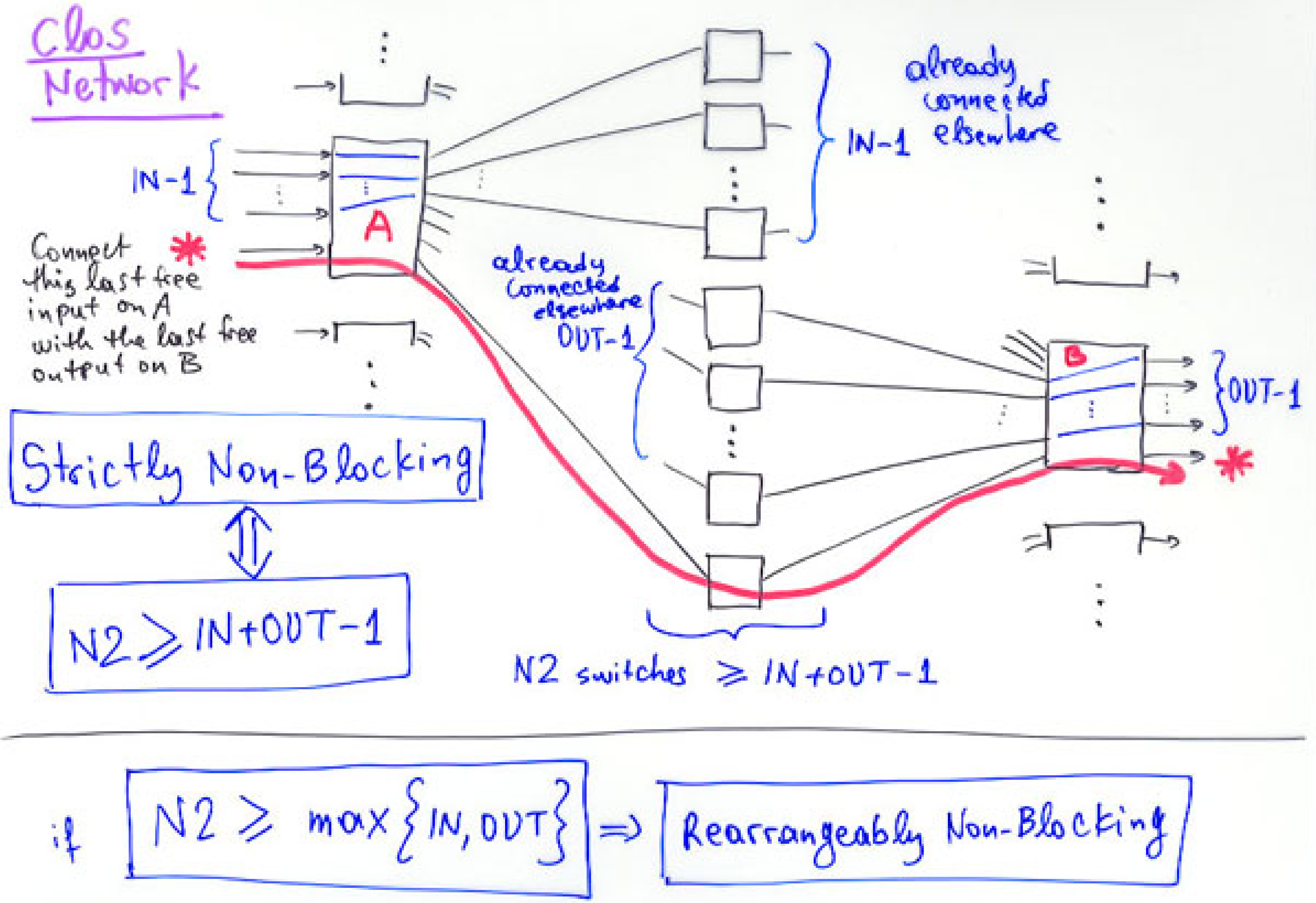
# Clos Networks (Generalization of Benes Networks)




5-parameter Network: (IN, N1, N2, N3, OUT)

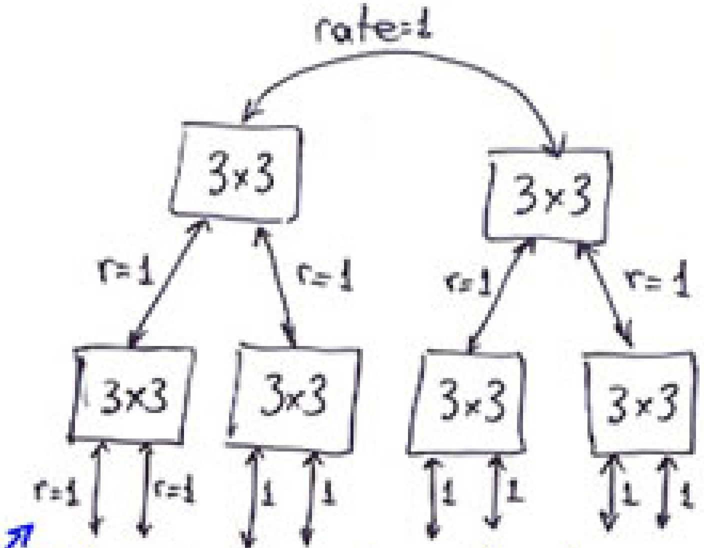
this example: the (3, 4, 5, 4, 3) Clos Network

usually: IN = OUT, N1 = N3

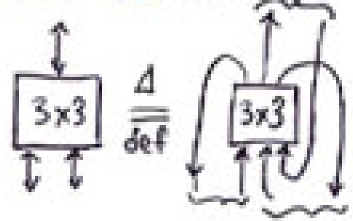

Clos Network
IN-1
Connect this last free input on A with the last free output on B
A
already connected elsewhere
IN-1
already connected elsewhere
OUT-1
B
OUT-1
Strictly Non-Blocking
N2 ≥ IN+OUT-1
N2 switches ≥ IN+OUT-1
if N2 ≥ max{IN, OUT} ⇒ Rearrangeably Non-Blocking

# FAT TREES

Customizable amounts of internal blocking, while exploiting locality of traffic, with option for non-blocking operation in a configuration similar to a folded Benes network.

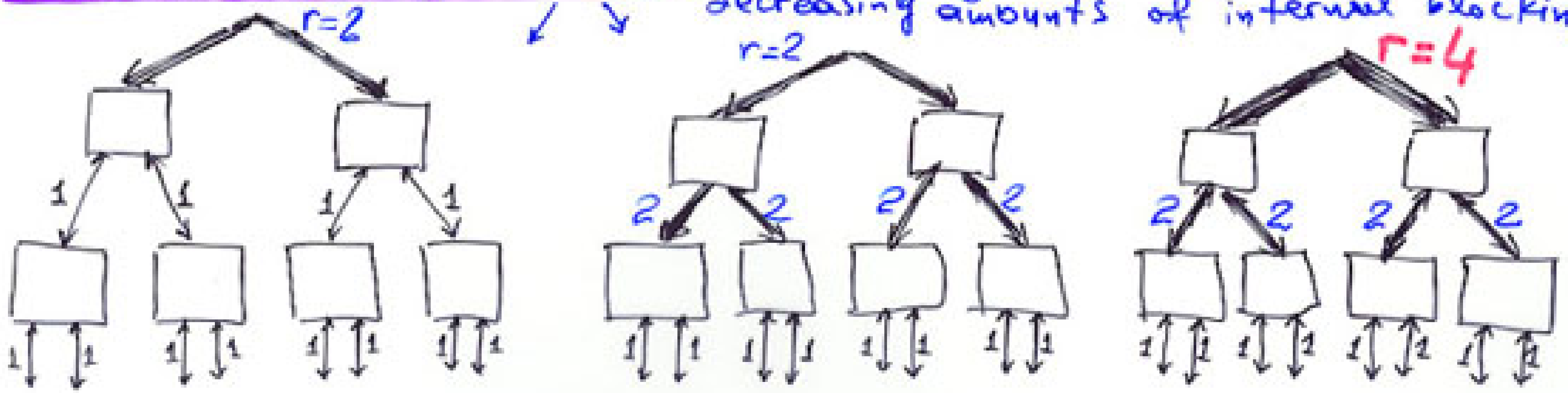



Plain Tree:
- Low Cost
- Lots of Internal Blocking, unless traffic is very local




The routes in a tree do not all have the same length, unlike what happens in Benes/banyan networks.

Conceptual View of Fat Trees of increasing cost and decreasing amounts of internal blocking:



# Fat Tree Implementation: ...a la folded Benes...