Multistage switching fabrics - Part I

Paolo Giaccone

Version: September 30, 2016
Outline

- Introduction
- Complexity and performance of switches
- Space switching
- Lee’s method
Introduction

Switching contexts

- packet switching (as in the Internet)
- circuit switching (as in the traditional telephone network)

Switching scenarios for different space scaling

- among different processing modules inside a chip
- among chips on the same linecard
- among hosts in a layer-2 network (switch)
- among servers in a data center
- among networks in a layer-3 network (router)
Reference architecture for space switching

- crossbar $N \times M$
  - each internal port may switch an aggregation of external ports (line-grouping)
- best performance
- simple control
- high implementation complexity

![Diagram of crossbar switch architecture](image-url)
Implementation complexity

- number of basic switching modules
- number of crosspoints
  - related to the number of logical gates and the area on a chip
  - for crossbar: $C(N \times M) = NM$
  - for symmetric crossbar: $C(N) = N^2$
- many other cost functions, depending on the particular technology used for implementation
  - scalability and modularity
  - power consumption
  - reliability
  - switch control and management
  - 2D/3D layout
Performance

- under *admissible* (i.e., non conflicting) switching requests (circuits or packets)
- non blocking: any input can be always connected to an idle output
  - strictly non blocking (SNB): any new connection does not change the pre-existing connections
  - rearrangeable (REAR): any new connection *may* change some pre-existing connections
- crossbar is SNB by construction
- SNB implies REAR but not viceversa
Space switching

- Traffic support
  - Unicast
  - Multicast

- Multistage networks
  - modules
  - stages
Full interconnections among stages

- Two stage switch, with **full interconnections** among the I-stage modules and the II-stage modules

- Possible (equivalent) graphical descriptions:
Lee’s method
Lee’s method

- *approximated* blocking analysis of multistage networks

- assumptions:
  - traffic uniformly distributed among inputs and outputs
  - random routing policy to distribute uniformly the traffic across the modules and links
  - *independence* of the busy state among all the links

- evaluate the blocking probability “seen” by a new circuit to be established, in function of the offered load
Lee’s method

- let $\rho$ be the average load of each input (i.e., the fraction of time the input is busy):
  \[ \rho \in [0, 1] \]

- let $\rho_{tot} = N\rho$ be the total load of each input: $\rho_{tot} \in [0, N]$ Erlang

- examples
  - in a $10 \times 10$ telephone switch, each input receives 6 calls/hour and each call lasts on average 3 minutes; then $\rho = 0.3$ and the total load is $\rho_{tot} = 3$ Erlang
  - in a $10 \times 10$ packet switch, with ports at 100 Mbps, each input receives on average $10^3$ pkt/s, each of size 1500 bytes; then $\rho = \frac{1500 \times 8 \times 10^3}{10^8} = 0.12$ and $\rho_{tot} = 1.2$ Erlang
Lee’s method for two stages

- symmetric network, with $N = pq$ ports
- $\rho$ is the average input load

\[
a = \frac{\rho N}{q^2} \quad \Rightarrow \quad P_b = \rho \frac{p}{q}
\]

\[C = 2qN\]

Note that for $\rho \geq \frac{q}{p}$, $P_b = 1$. 
Lee’s method for two stages

- symmetric network, with $N = pq$ ports
- $\rho$ is the average input load
- $l$ multiple edges

\[ a = \frac{\rho N}{lq^2}, \quad b = a^l \quad \Rightarrow \quad P_b = \left(\frac{\rho \frac{p}{lq}}{lq^2}\right)^l \]

\[ C = 2lqN \]

Note that for $\rho \geq \frac{lq}{p}$, $P_b = 1$. 
Lee’s method for three stages

- symmetric network, with \( N = pq \) ports
- \( \rho \) is the average input load

\[
a = \frac{\rho N}{qr}, \quad b = 1 - (1-a)^2 = 2a - a^2, \quad c = b^r = a^r(2-a)^r \quad \Rightarrow \quad P_b = \rho^r \left[ \frac{2N}{qr} - \frac{\rho N^2}{q^2r^2} \right]^r
\]

\[
C = 2rN + rq^2
\]

Note that for \( \rho \geq \frac{r}{p} \), \( P_b = 1 \).
Design comparison

\[ N = 1024, \rho = 0.01 \]
Design comparison

$N = 1024, \rho = 0.01$
Multistage switching fabrics - part II

Paolo Giaccone
Outline

- Clos network
  - strictly non blocking: Clos theorem
  - rearrangeable: Slepian Duguid theorem
  - Paull’s matrix and Paull’s algorithm

- Recursive construction
  - Benes network \((p = 2)\), looping algorithm
  - \(p = \sqrt{N}\)

- Self routing
  - Banyan networks
Clos networks
Clos networks

- three stage networks
  - $m_i$: number of inputs for modules at stage $i$
  - $n_i$: number of outputs for modules at stage $i$
  - $r_i$: number of modules at stage $i$
  - $M_i = \{1, 2, \ldots, r_i\}$ is the set of modules identifiers belonging to $i$-th stage

- Exactly one link between two modules in successive stages
  - $r_1 = m_2$, $r_2 = n_1 = m_3$, $r_3 = m_2$
Clos network

$N \times M$ Clos network with $N = m_1 r_1$ and $M = r_3 n_3$
SNB Clos networks

Clos Theorem:

A Clos network is SNB if and only if the number of second stage switches \( r_2 \) satisfies:

\[
r_2 \geq m_1 + n_3 - 1
\]

In particular, a symmetric network with \( m_1 = n_3 = n \) is SNB if and only if

\[
r_2 \geq 2n - 1
\]

Proof: Assume that module \( i \) of the I-stage should be connected to module \( j \) of the III-stage. Hence, a new symbol should be added in \( P_{ij} \) of Paull's matrix \( P \). In the worst case, there are already \( m_1 - 1 \) symbols in the \( i \)-th row of \( P \) and \( n_3 - 1 \) symbols in the \( j \)-th column. They are all distinct. Hence, to find a new symbol available in the II-stage, it should be

\[
r_2 > (m_1 - 1) + (n_3 - 1) \text{ which implies } r_2 \geq m_1 + n_3 - 1.
\]
Complexity of a SNB Clos network

- consider a symmetric Clos network, with $m_1 = n_3 = p$, $r_1 = r_3 = q$ with $N = pq$
- thanks to Clos Theorem, the smallest Clos network is built with $r_2 = 2p - 1$
- total complexity:

$$C_{SNB}(N) = qC(p \times (2p - 1)) + (2p - 1)C(q \times q) + qC((2p - 1) \times p) = (2p - 1)(2pq + q^2)$$

- approximated complexity (assume $r_2 = 2p$):

$$C_{SNB}(N) \approx qC(p \times (2p)) + 2pC(q \times q) + qC((2p) \times p) = 4p^2q + 2pq^2$$
Paull’s matrix

• describes the state of the active interconnections present in a Clos network (i.e., the switching configurations of all the II-stage modules)

• matrix $P = \begin{bmatrix} P_{ij} \end{bmatrix}$ of size $r_1 \times r_3$
  
  – $P_{ij}$ is a set of II-stage modules, i.e. $P_{ij} \subseteq M_2$
  
  – if $k \in P_{ij}$ means that II-stage module $k$ is connected to I-stage module $i$ and III-stage module $j$

  – feasibility conditions
    
    * each row with at most $m_1$ symbols
    * each column with at most $n_3$ symbols
    * each element with at most $\min\{m_1, n_3\}$ symbols
    * each $k \in M_2$ appears at most once for each row and for each column
Configuring a SNB Clos network

- when an input of I-stage module $i$ should be connected to an output of III-stage module $j$, find any II-stage module $k$ such that the connections $i \rightarrow k$ and $k \rightarrow j$ are both free
  - such connection always exists thanks to the Clos theorem
  - in Paull's matrix $P$, this operation corresponds to find any available symbol in both $i$-th row and $j$-th column
Rearrangeable non-blocking Clos networks

Slepian-Duguid Theorem:

A Clos network is REAR if and only if the number of second stage switches \( r_2 \) satisfies:

\[
r_2 \geq \max\{m_1, n_3\}
\]

In particular, a symmetric network with \( m_1 = n_3 = n \) is SNB if and only

\[
r_2 \geq n
\]

Proof: It will be proved using the Birchkoff von Neumann theorem, later in the course
Complexity of a REAR Clos network

- consider a symmetric Clos network, with $m_1 = n_3 = p$, $r_1 = r_3 = q$ with $N = pq$
- thanks to Slepian Duguid Theorem, the smallest Clos network is built with $r_2 = p$
- total complexity:

$$C_{REAR}(N) = qC(p \times p) + pC(q \times q) + qC(p \times p) = 2qp^2 + pq^2$$
Clos complexity comparison

By setting \( q = \frac{N}{p} \) in the formulas of the Clos networks complexity:

\[
C_{SNB} = (2p - 1) \left(2N + \frac{N^2}{p^2}\right) \approx 4pN + \frac{2}{p}N^2
\]

\[
C_{REARR} = 2pN + \frac{1}{p}N^2
\]

and hence,

\[
C_{SNB}(N) = \frac{2p - 1}{p} C_{REAR}(N)
\]

which means:

\[
C_{REAR}(N) \leq C_{SNB}(N) < 2C_{REAR}(N)
\]

Note that, to be advantageous with respect to the crossbar, it should be:

\[
C_{REAR}(N) < N^2 \quad C_{SNB}(N) < N^2
\]
Clos complexity comparison

Number of crosspoints vs. $p$ for different values of $N$: SNB $N=10$, REAR $N=10$, SNB $N=100$, REAR $N=100$, SNB $N=1000$, REAR $N=1000$, SNB $N=10000$, REAR $N=10000$.
Clos complexity comparison
Minimum complexity for REARR Clos network

- minimum of $C_{\text{REARR}}$ obtained for $\hat{p}$:

$$\frac{\partial C_{\text{REARR}}}{\partial p} = 2N - \frac{N^2}{p^2} = 0 \quad \Rightarrow \quad \hat{p} = \sqrt{\frac{N}{2}}$$

Hence, the minimum complexity is:

$$C_{\text{REARR}}^{\text{opt}} = 2\sqrt{2N}\sqrt{N} = \Theta(N\sqrt{N})$$

- for any $N > 8$, $C_{\text{REARR}}^{\text{opt}} < C_{\text{crossbar}} = N^2$

- for $p = 1$, the Clos network degenerates into a $N \times N$ crossbar;

$$C_{\text{REAR}}(p = 1) = N^2$$

- for $p = N$, the Clos network degenerates into two tandem $N \times N$ crossbars;

$$C_{\text{REAR}}(p = N) = 2N^2$$
Configuring a REAR Clos network

- **Paull’s algorithm**
  - incremental algorithm, used to add one connection at one time and reconfigure the network if needed
  - will be also used to support rate guarantees in input queued switches
  - *(Paull’s Theorem)* for each new connection, the number of connections needed to be rearranged is at most $\min\{r_1, r_3\} - 1$
  - for each new connection, the number of II-stage modules to reconfigure is at most two
Paull’s algorithm

- Given Paull’s matrix \( P = [P_{ij}] \) and a new connection to add in \( P_{ij} \); two cases are possible
  - it exists a II-stage module \( a \) which is available in both row \( i \) and column \( j \) of \( P \); hence, use module \( a \) for the new connection, without any rearrangement: \( P_{ij} = P_{ij} \cup a \)
  - otherwise, there should be two II-stage modules \( a \) and \( b \) such that \( a \) is available in row \( i \), and \( b \) is available in column \( j \) of \( P \). Find an \((a, b)\)-path (or a \((b, a)\)-path) starting from \( P_{ij} \). Now swap \( a \) with \( b \) in such path, and use \( a \) (or \( b \) for the \((b, a)\)-path) for the new connection: \( P_{ij} = P_{ij} \cup a \).

\[\begin{align*}
? &\quad a \\
\text{(a,b)-path} &\quad (a,b)\text{-path} \\
b &\quad a \\
\text{(b,a)-path} &\quad (b,a)\text{-path} \\
b &\quad a \\
\end{align*}\]
Recursive construction of switching networks
Recursive construction of switching networks

- main idea to exploit recursively
  - to build a REAR Clos network, use a REAR Clos network for each module
  - to build a SNB Clos network, use a SNB Clos network for each module
- many ways to factorize the network
  - for small complexity, keep small $p$

\[
C_{REAR}(N) = 2qp^2 + pq^2 \quad C_{SNB}(N) = (2p - 1)q(2p + q)
\]
\[
C_{REAR}(N, p = 2) = \frac{N^2}{2} + 4N \quad \text{vs.} \quad C_{SNB}(N, p = 2) = \frac{3N^2}{4} + 6N
\]
- for keeping the same “aspect ratio”, use $p = \sqrt{N}$
Benes network

- Clos network, REAR, recursively factorized with $p = 2$, exploiting only $2 \times 2$ modules
- $N = 2^n$ for some $n$
Example of Benes networks
Benes network complexity

The number of crosspoints satisfies:

\[ C(N) = NC_2 + 2C\left(\frac{N}{2}\right) = kNC_2 + 2^kC\left(\frac{N}{2^k}\right) \quad \text{for} \quad k = 0, \ldots, \log_2 N - 1 \]

Now, by setting \( k = \log_2 N - 1 \) and considering \( C_2 = 4 \):

\[ C(N) = N(\log_2 N - 1)C_2 + \frac{N}{2}C_2 = 4N \log_2 N - 2N \]

The number of stages satisfies:

\[ S(N) = 2 + S\left(\frac{N}{2}\right) = 2k + S\left(\frac{N}{2^k}\right) \quad \text{for} \quad k = 0, \ldots, \log_2 N - 1 \]

and again, by setting \( k = \log_2 N - 1 \):

\[ S(N) = 2 \log_2 N - 1 \]
Benes network configuration

Two algorithms:

- Paull’s algorithm applied recursively
- Looping algorithm
  - equivalent to Paull’s algorithm using a particular sequence of switching requests
  - all the switching requests should be known in advance to avoid reconfigurations
Master method for recurrence equations

- Landau notation for $f(n), g(n) > 0$
  - $f(n) = \Theta(g(n))$ means that $\exists c, c' > 0, n_0$ s.t. $\forall n \geq n_0$:
    $$cg(n) \leq f(n) \leq c'g(n)$$
  - $f(n) = O(g(n))$ means that $\exists c > 0, n_0$ s.t. $\forall n \geq n_0$: $f(n) \leq cg(n)$
  - $f(n) \sim g(n)$ means that $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 1$

- Master method to solve $T(n) = aT(n/b) + f(n), a \geq 1, b \geq 1$
  - if $\exists \epsilon > 0$ s.t. $f(n) = O(n^{\log_b a - \epsilon})$, then
    $$T(n) = \Theta(n^{\log_b a})$$
  - if $f(n) = \Theta(n^{\log_b a})$, then
    $$T(n) = \Theta(n^{\log_b a \log_2 n})$$
Clos networks, factorized recursively with factor 2

- REAR Clos network, factorized recursively, $p = 2$ (i.e., Benes network)
  - $C(N) = NC_2 + 2C(N/2)$
  - using master method, $a = b = 2$, then $f(n) = \Theta(N)$; hence,
    $$C(N) = \Theta(N \log_2 N)$$

- SNB Clos network, factorized recursively, $p = 2$
  - $C(N) = 2NC_2 + 3C(N/2)$
  - using master method, $a = 3$, $b = 2$, then $f(n) = \Theta(N) = O(n^{\log_2 3-\epsilon})$ with $\epsilon = 0.5$; hence,
    $$C(N) = \Theta(N^{\log_2 3}) \approx \Theta(N^{1.58})$$
REAR Clos network, factorized recursively with factor $\sqrt{N}$

For convenient factorization, assume $N = 2^n$ and $n = 2^k$.

$$C(N) = 3\sqrt{N}C(\sqrt{N}) = 3 \cdot 2^{n/2}C\left(2^{n/2}\right) = 3^k 2^{n/2 + n/2^2 + \ldots + n/2^k} C\left(2^{n/2^k}\right)$$

If we set $k = \log_2 n$, since $1/2 + 1/2^2 + \ldots + 1/2^k \approx 1$ for large $k$ (i.e., large $N$)

$$C(N) \approx 3^{\log_2 n} 2^n C(2) = n^{\log_2 3} N C(2) = 4N (\log_2 N)^{1.58}$$
SNB Clos network, factorized recursively with factor $\sqrt{N}$

For convenient factorization, assume $N = 2^n$ and $n = 2^k$. For a better layout, we assume
that $r_2 = 2\sqrt{N}$ and then:

$$C(N) = \sqrt{NC}(\sqrt{N} \times 2\sqrt{N}) + 2\sqrt{NC}(\sqrt{N}) + \sqrt{NC}(2\sqrt{N} \times \sqrt{N})$$

Since $C(\sqrt{N} \times 2\sqrt{N}) = 2C(\sqrt{N})$,

$$C(N) = 6\sqrt{NC}(\sqrt{N}) = 6 \cdot 2^{n/2}C\left(2^{n/2}\right) = 6^k \cdot 2^{n/2+n/2^2+...+n/2^k}C\left(2^{n/2^k}\right)$$

If we set $k = \log_2 n$, since $1/2 + 1/2^2 + ... + 1/2^k \approx 1$ for large $k$ (i.e., large $N$)

$$C(N) \approx 6^{\log_2 n}2^nC(2) = n^{\log_2 6}NC(2) = 4N(\log_2 N)^{2.58}$$
Recursive factorization - summary

- $p = 2$
  - REAR $\Rightarrow C(N) = 4N \log_2 N$ (Benes)
  - SNB $\Rightarrow C(N) = \Theta(N^{1.58})$

- $p = \sqrt{N}$
  - REAR $\Rightarrow C(N) = 4N(\log_2 N)^{1.58}$
  - SNB $\Rightarrow C(N) = 4N(\log_2 N)^{2.58}$
Banyan networks

- self-routing $N \times N$ switch
  - header of the packet drives the routing path
- complexity $\Theta(N \log_2 N)$
- unique path from each input to each output
- based on the Benes network
Examples of Banyan networks

Baseline network

Banyan network

Shuffle exchange (Omega) network

Flip (inverse shuffle exchange) network
Blocking in Banyan networks

- **Property:** if self-routing addresses satisfy both conditions:
  - strictly monotone outputs, i.e. output destinations are increasing at the inputs
  - compact monotone inputs, i.e. no idle inputs between any two active inputs
then the self-routing is non-blocking

- in general, Banyan networks are blocking
  - number of $2 \times 2$ modules in a Banyan network: $(N/2) \log_2 N$
  - number of possible states in a Banyan network: $S_{\text{Banyan}} = 2^{(N/2) \log_2 N}$
  - taking logarithm:
    \[
    \log_2 S_{\text{Banyan}} = (N/2) \log_2 N \sim \frac{1}{2} \log_2 N!
    \]

- the probability that a random input-output permutation is non-blocking is $2^{-(N/2) \log_2 N}$
  which goes to zero very quickly by increasing $N$
Batcher-Banyan network

Two switching phases:

- (1) self-sorting Batcher network
  - transform any switching request into a non-blocking switching request for Banyan network

- (2) self-routing Banyan network

- final complexity = $\Theta(N (\log_2 N)^2)$