Throught the use of OOB programming in `Python` , a model has been built in order to mimic the execution of the **broadcast** and **reduce** algorithms presented in section **??**. The core idea has been to build an "artificial" architecture of the EPYC nodes used in the experiment that was able to simulate the most important aspects in the latency measurements.
The model (organized in a `Python` module) is in fact a collection of classes and methods that accurately reproduce the internal architecture of the EPYC nodes used for the benchmark and some of the behaviours of the MPI library. For instance, this module offers the possibility of instantiating an object of class `Process` that can be binded to a specific `Core` , located in one of the 2 `Socket` objects, enclosed in a `Node` class. This seemingly complex structure is in truth notthing else than a simplified reproduction of the EPYC node's internal architecture. `Process` objects are then given a `send` method that aims to replicate the execution of a NBFT as explained in section **??**. In fact, this method takes in input a list of receiving processesses and a message size, computes the relative positions of all the other processes with respect to the sender to understand which communication channels to use, and returns the latency of the NBFT estimated using equation **??**, using the fitted point-to-point latencies and the discrete function $\widehat{\gamma}^c(P,m)$ obtained in equation **??**.
In other words, this model can accurately reproduce the execution time of an arbitrarily large NBFT for any message size. According to equations **??** and **??**, this can then be used to also estimate the execution time of the **chain** and **binary tree** algorithms. However, a problem that this model had to face was the situation in which more than one communication channel was used in the same NBFT. Figures **??**b and **??**c show this exact situation where links in red and green are used to indentify two different communication channels. As explained, this model takes into account 4 possible communication channels, ordered according to:

$$c_i \geq c_j \quad \Leftrightarrow \quad T_{p2p}^{c_i}(m) \geq T_{p2p}^{c_j}(m) \quad , \forall m,\ \forall i,j \in \{0,1,2,3\} \tag{1}$$

Therefore, this model assumes that:

$$T_{p2p}^{c_i}(m) = Q_{i,j}(m) \cdot T_{p2p}^{c_j}(m) \quad , \forall m,\ \forall i,j \in \{0,1,2,3\} |\ i > j \tag{2}$$

where $Q_{i,j}(m)$ is a platform-dependent but algorithm-independent parameter ($Q_{i,j}(m) \geq 1$) representing the *ratio of delayes* between any two communication channels c_i and c_j. This parameter is easily estimated using the fitted values for the point-to-point communication in equation **??** as follows:

$$Q_{i,j}(m) = \frac{\widehat{T}_{p2p}^{c_i}(m)}{\widehat{T}_{p2p}^{c_j}(m)} \quad , \forall m,\ \forall i,j \in \{0,1,2,3\} |\ i > j \tag{3}$$

Therefore, the execution time T_{NBFT} of a NBFT **A** using multiple channels is calculated as the time $T_{NBFT}^{c_{\max}}$ of the NBFT **B** that only uses the highest-order channel, obtained from **A** by formal replacement of of each group of $Q_{\max,j}(m)$ point-to-point communications in channel c_j with a single point-to-point communication in channel $c_{\max}$, where $c_{\max}$ is the highest-order channel used in **A** while c_j can be any lower-order channel. Formally, by labelling with N_i the number of point-to-point communication through channel c_i, this yields the following equation:

$$T_{NBFT}(P,m) = \begin{cases} T_{NBFT}^{c_0}(P,m) & \text{if } c_{\max} = c_0 \\ T_{NBFT}^{c_1}(N_1 + \lfloor \frac{N_0}{Q_{1,0}(m)} \rfloor + 1, m) & \text{if } c_{\max} = c_1 \\ T_{NBFT}^{c_2}(N_2 + \lfloor \frac{N_1}{Q_{2,1}(m)} \rfloor + \lfloor \frac{N_0}{Q_{2,0}(m)} \rfloor + 1, m) & \text{if } c_{\max} = c_2 \\ T_{NBFT}^{c_3}(N_3 + \lfloor \frac{N_2}{Q_{3,2}(m)} \rfloor + \lfloor \frac{N_1}{Q_{3,1}(m)} \rfloor + \lfloor \frac{N_0}{Q_{3,0}(m)} \rfloor + 1, m) & \text{if } c_{\max} = c_3 \end{cases} \tag{4}$$

where again, each $T_{NBFT}^{c_i}(P,m)$ is computed according to equation **??**, using the fitted values for the point-to-point communications and the *parallelization factor* found in section **??**. This concretely solves the problem of computing NBFT latencies with multiple concurrent communication channels and practically enables the model to predict the execution time of the collective operation algorithms in all possible scenarios.