Research Progress Seminar

A summary of Network-on-Chip benchmarks

Khanh N. Dang

Adaptive Systems Laboratory
The University of Aizu

January 11, 2017

Table of Contents

Introduction

Why do we select these benchmarks? Because these benchmarks can exploit the characteristics of NoC: parallelism and scalability.
Why not the realistic benchmarks? Because we aim the study the network only, not the system performance, therefore, selecting these benchmarks is enough for evaluating.

- Selecting and using benchmark are important for evaluating any system.
- Using benchmarks helps designer understand the performance of their own system and compare with others.
- This presentation aims to explain in details of the benchmarks which are used in OASIS.
  - Transpose
  - Uniform
  - Matrix-multiplication
  - Hotspot 10%
  - Realistic Traffic Pattern

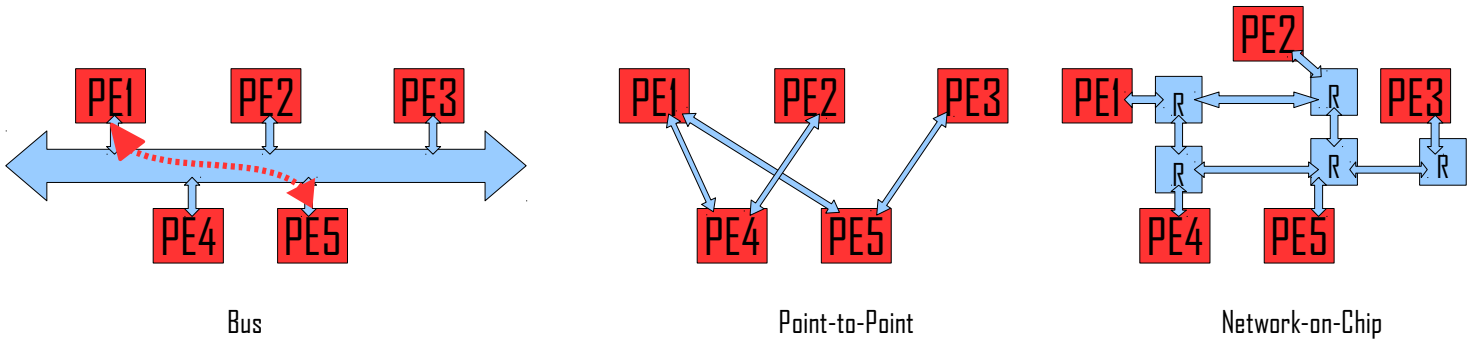
Network-on-Chip

Traditional on-chip communication methods

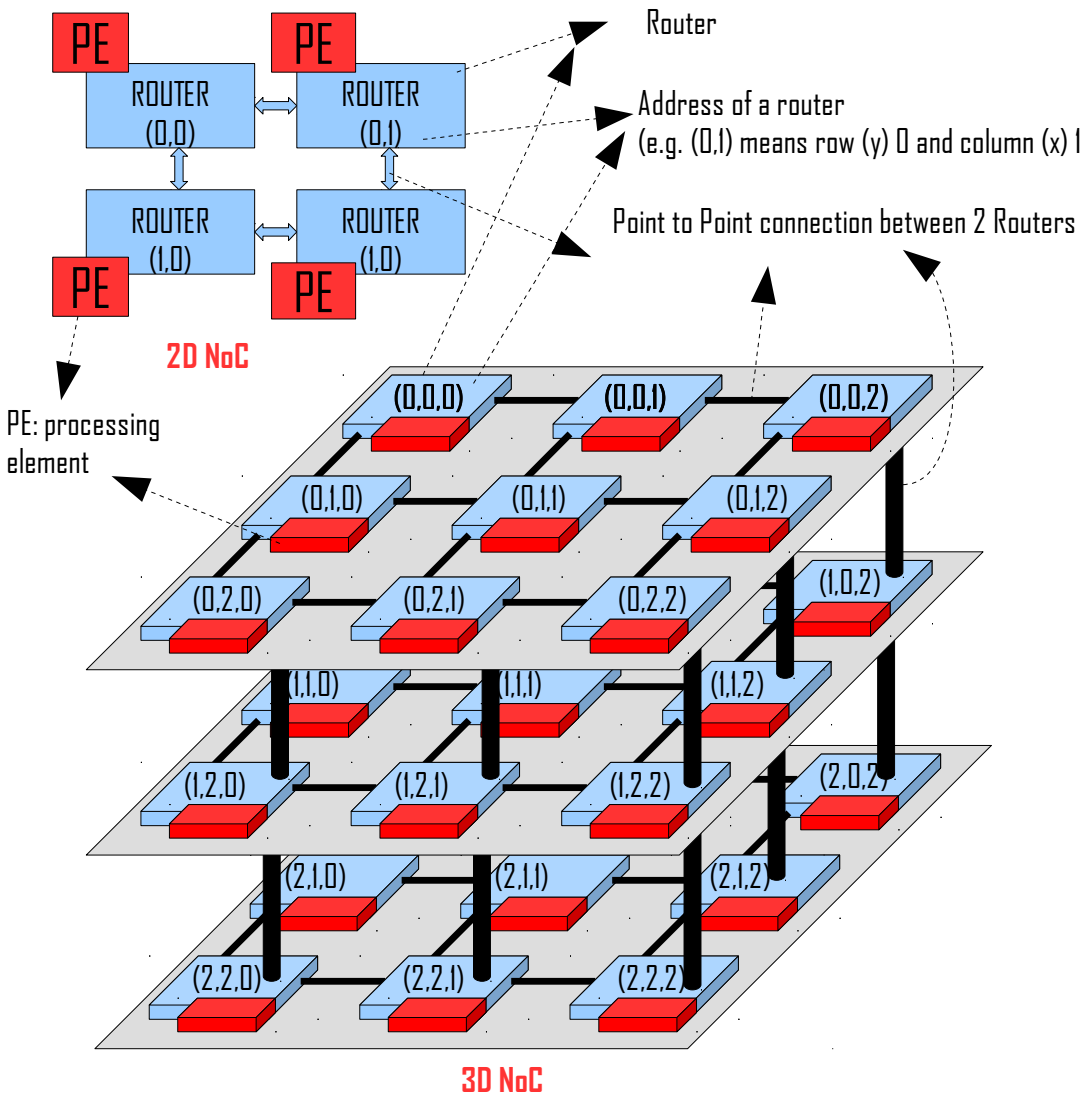
Bus: a shared connection between PE (Processing Elements). In a clock cycle, only one communication is performed.
Point-to-Point: direct connection between two PEs.

Network-on-Chip idea

PEs are attached to routers. A communication is done by routing the packets from the source node to the destination node.



NoC Annotation



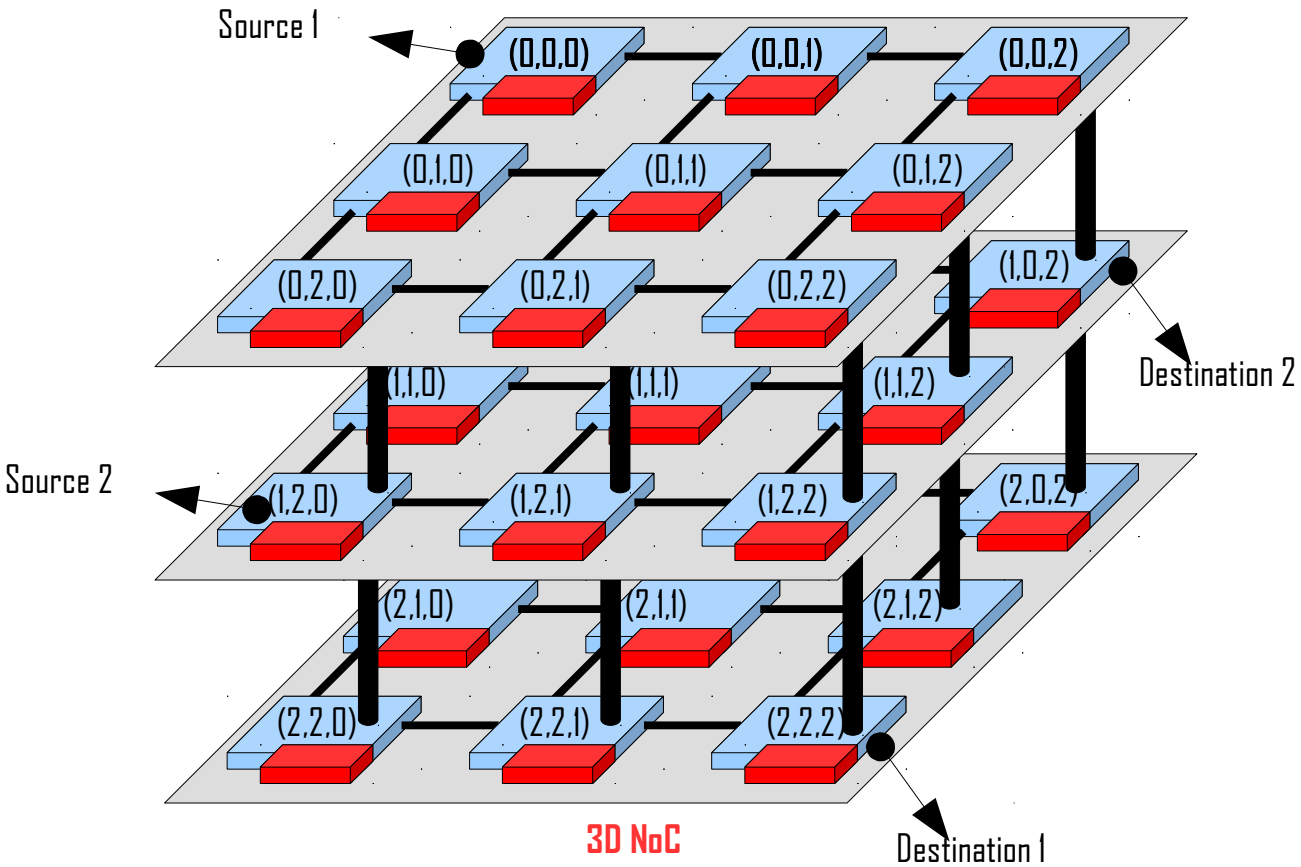
Detail of Benchmark

Benchmark	Description	Citation
Transpose	Each node (a,b,c) in a network with (X,Y,Z) sends packets to node (X-a, Y-b, Z-c)	[1]
Uniform	Each node in a network sends packets to all nodes	[2]
Matrix-multiplication	Performs C=A*B. Matrix A is stored in layer-1, is sent to layer-2 which has matrix B. The final values are accumulated in layer-3 as matrix C.	[3, 4]
Hotspot 10%	Each node in a network sends packets to all nodes. X (X=1 or 2 or more) nodes have additional 10% amount of traffic.	[5]
Realistic Traffic Pattern	Generate from task graphs which provide the connections (e.g: node A$\rightarrow$B) and the traffic (e.g: 100 packets).	[6, 7]

$\Rightarrow$ The following slides will explain these benchmarks in details.

Transpose

Transpose is a communication pattern which is based on tranposed matrix. Each nodes sends messages to a reversed dimension index node. A network with size (X,Y,Z). Node (a,b,c) in this network will send to node (X-a-1, Y-b-1, Z-c-1).



Transpose (cnt)

Why not (X-a,X-b,Z-c)? Consider a network with size (3,3,3). The lowest index is 0 and the highest index is 2. If we use (X-a, Y-b, Z-c), router with index 0 will send to router with index 3 (the correct index is 2).
What if the index start from 1 instead 0? Consider a network with size (3,3,3). The lowest index is 1 and the highest index is 3. If we use (X-a, Y-b, Z-c), router with index 1 will send to router with index 2 (the correct index is 3). $\Rightarrow$ In this case, the correct index is (X+1-a, Y+1-b, Z+1-c).
Characteristic:

- Consists of long and short communication hops. For example: A network of $3 \times 3 \times 3$ has 7 hopes connections (e.g (0,0,0) $\rightarrow$ (2,2,2)) and 1 hopes connections (e.g (1,1,1) $\rightarrow$ (1,1,1)). ▸ Matlab code
- Bottle-neck: the vertical connection is shared between several connections. A network of $X \times Y \times Z$ has $floor(Z/2)$ pairs of node share a vertical communication. For example: (X=6,Y=6,Z=6), vertical connection between (0,0,2) $\rightarrow$ (0,0,3) is shared between 3 pairs: (0,0,2) $\rightarrow$ (0,0,3),(0,0,1) $\rightarrow$ (0,0,4) and (0,0,0) $\rightarrow$ (0,0,5).

Transpose: Communication Length Distribution

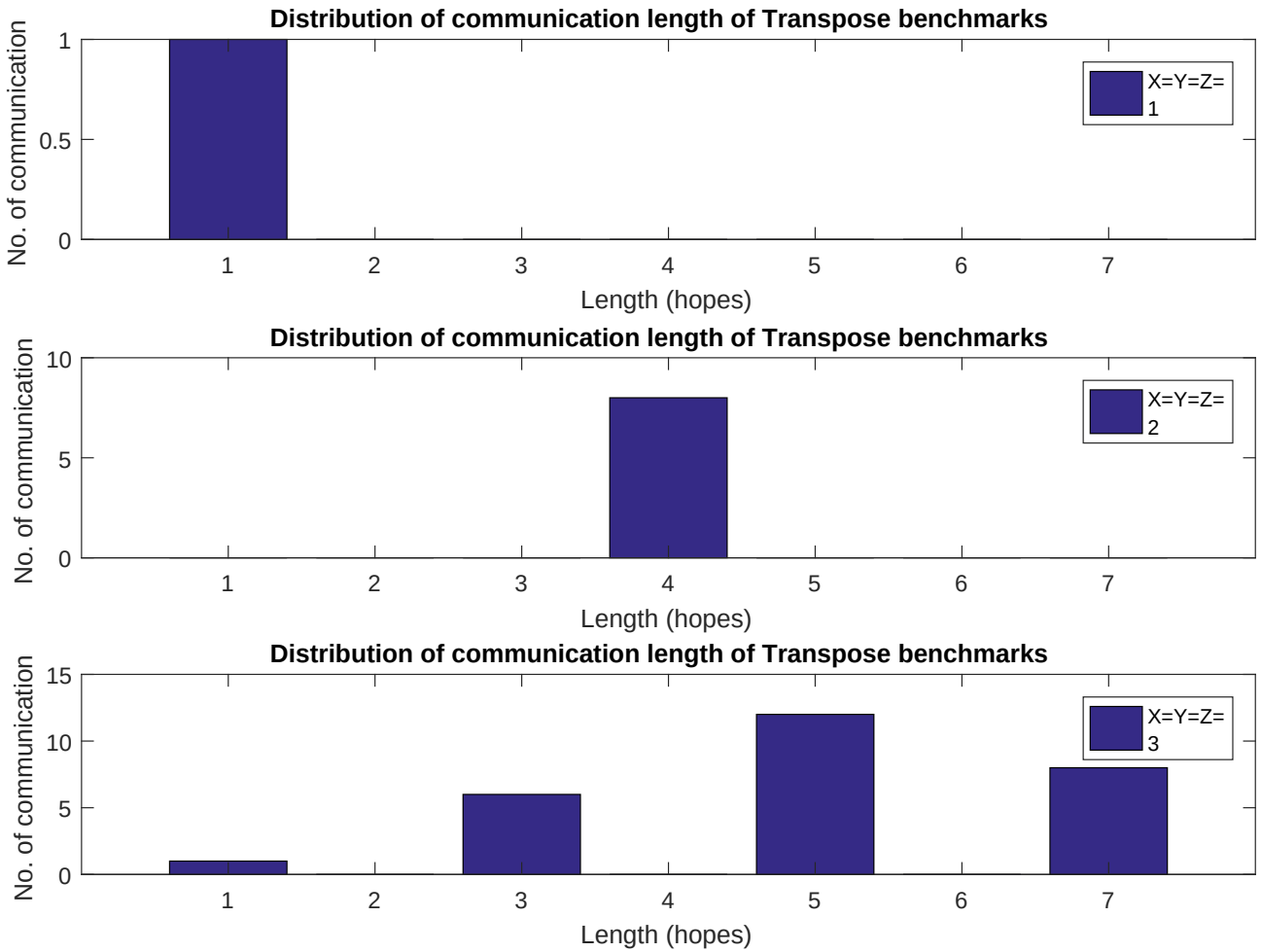


Figure 1: Distribution of communication length. ▸ Matlab code

Transpose: Communication Length Distribution

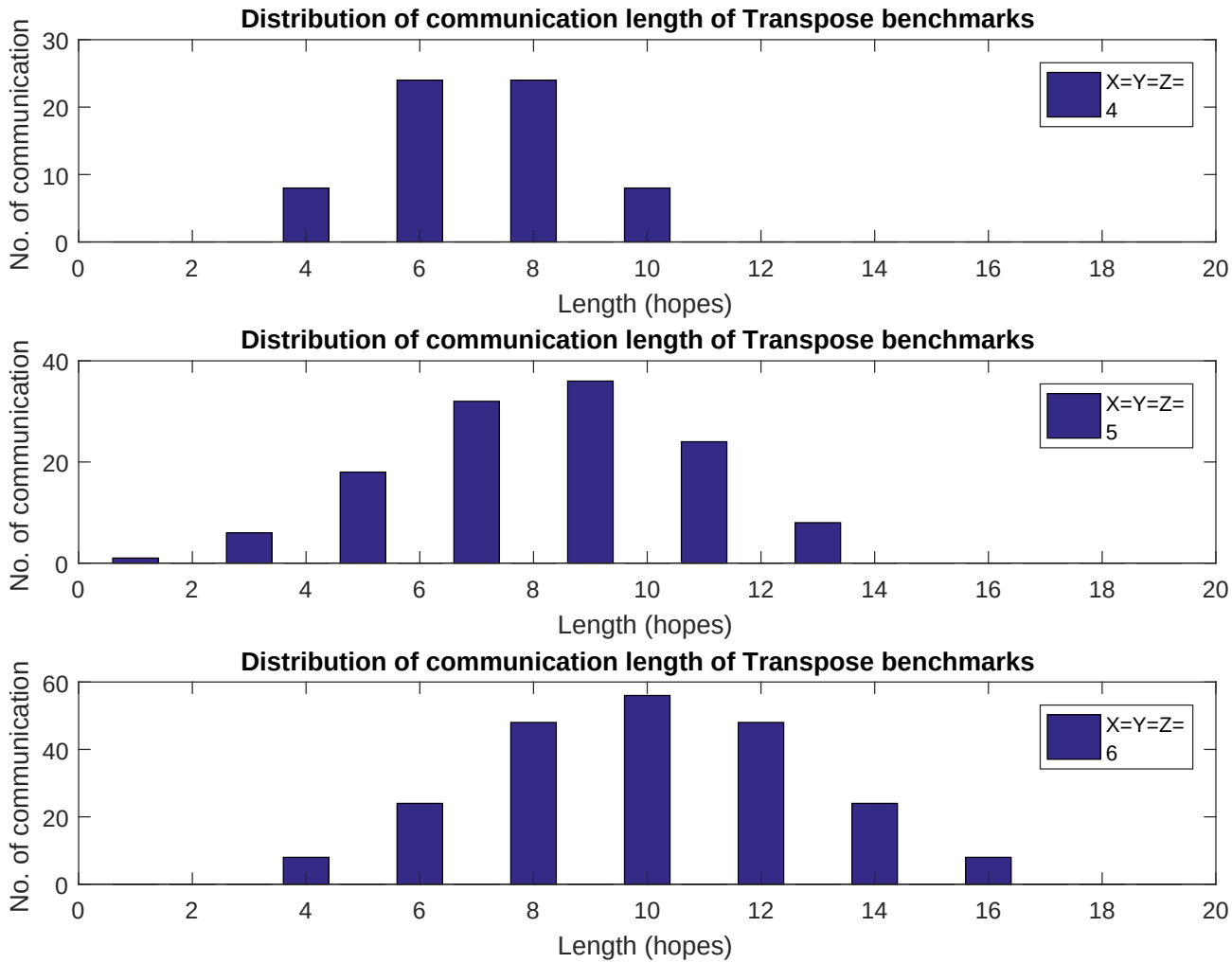


Figure 2: Distribution of communication length.

Transpose: Communication Length Distribution

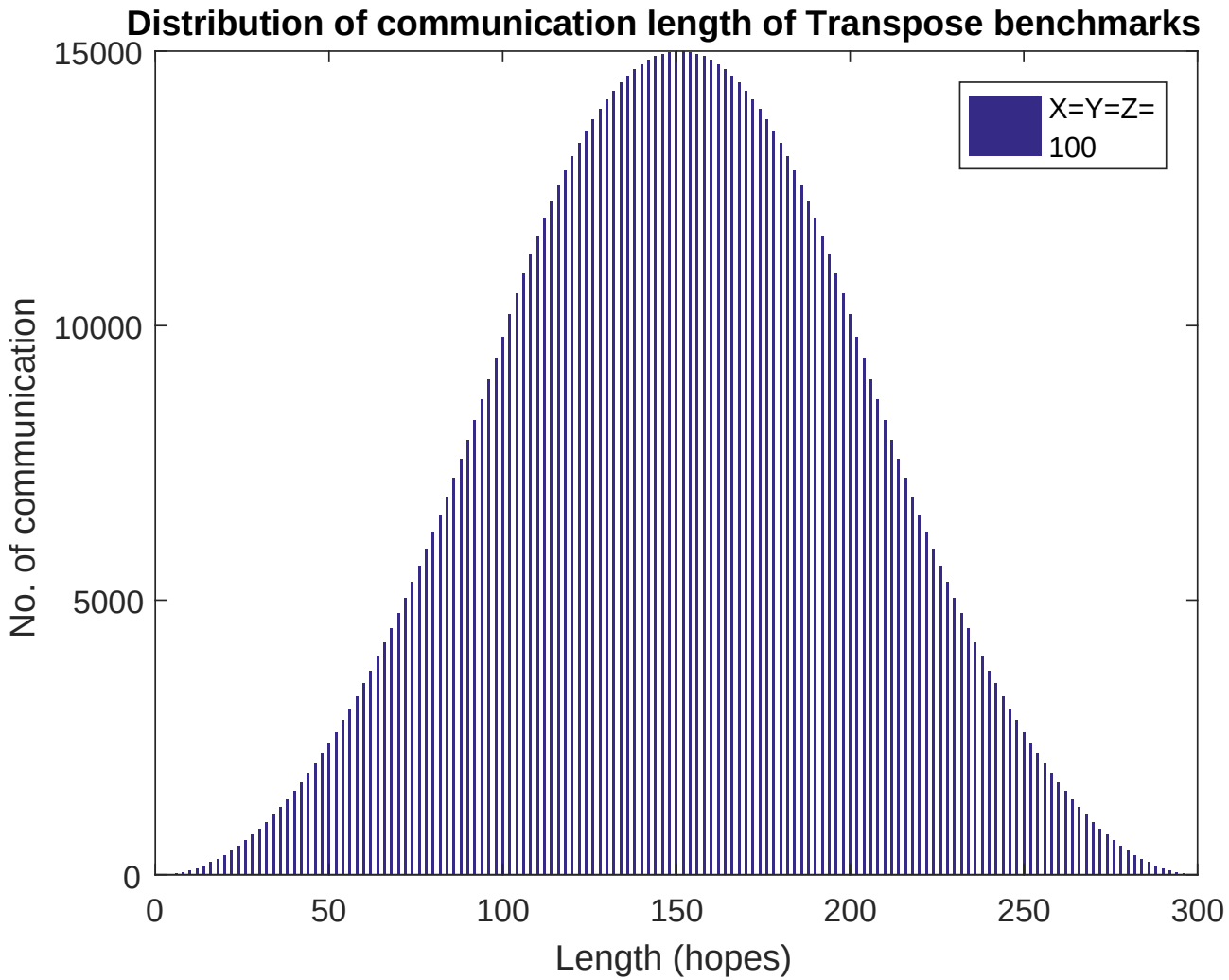
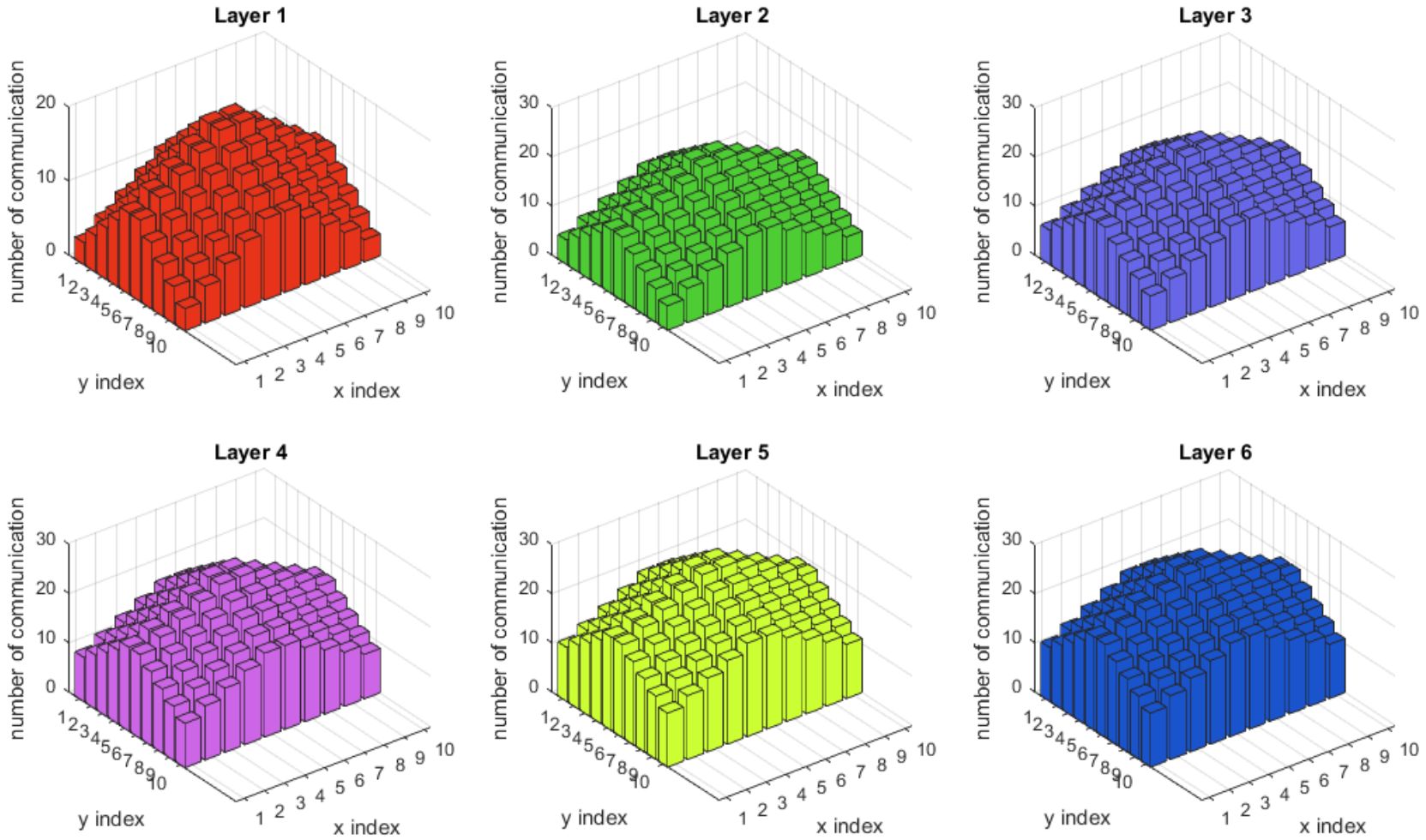


Figure 3: Distribution of communication length. Network size: (100,100,100)

Transpose: Bottle-neck Analysis / network (10,10,10)



Note: This analysis is based on XYZ routing. Each bar represents for the communication of one router.

Transpose algorithm

Algorithm 1: Transpose Algorithm.

// Network
Input: *Network*(X, Y, Z)
// Amount of data for each communication
Input: D
// Communication set
Output: $C = \{c_i : (source \rightarrow destination, \text{amount of data})\}$
1 **foreach** *node (a,b,c) in Network*(X, Y, Z) **do**
2 | **add** $((a, b, c) \rightarrow (X - a - 1, Y - b - 1, Z - c - 1)$, D packets) **to** C
3 **end**
4 **return** C

Uniform

Uniform is one of standard benchmarks for Network-on-Chip. Each nodes send messages to every node with an equal probability. For example: a network with size (3,3,3). A node (a,b,c) sends the same amount of packets to all node (0,0,0) $\rightarrow$ (2,2,2). In a network (X,Y,Z), probability of source node (a,b,c) is:

$$P_{S==(a,b,c)} = 1/(X \times Y \times Z) \tag{1}$$

Probability of destination node (m,n,p) from source node (a,b,c):

$$P_{D==(m,n,p)} = 1/(X \times Y \times Z) \tag{2}$$

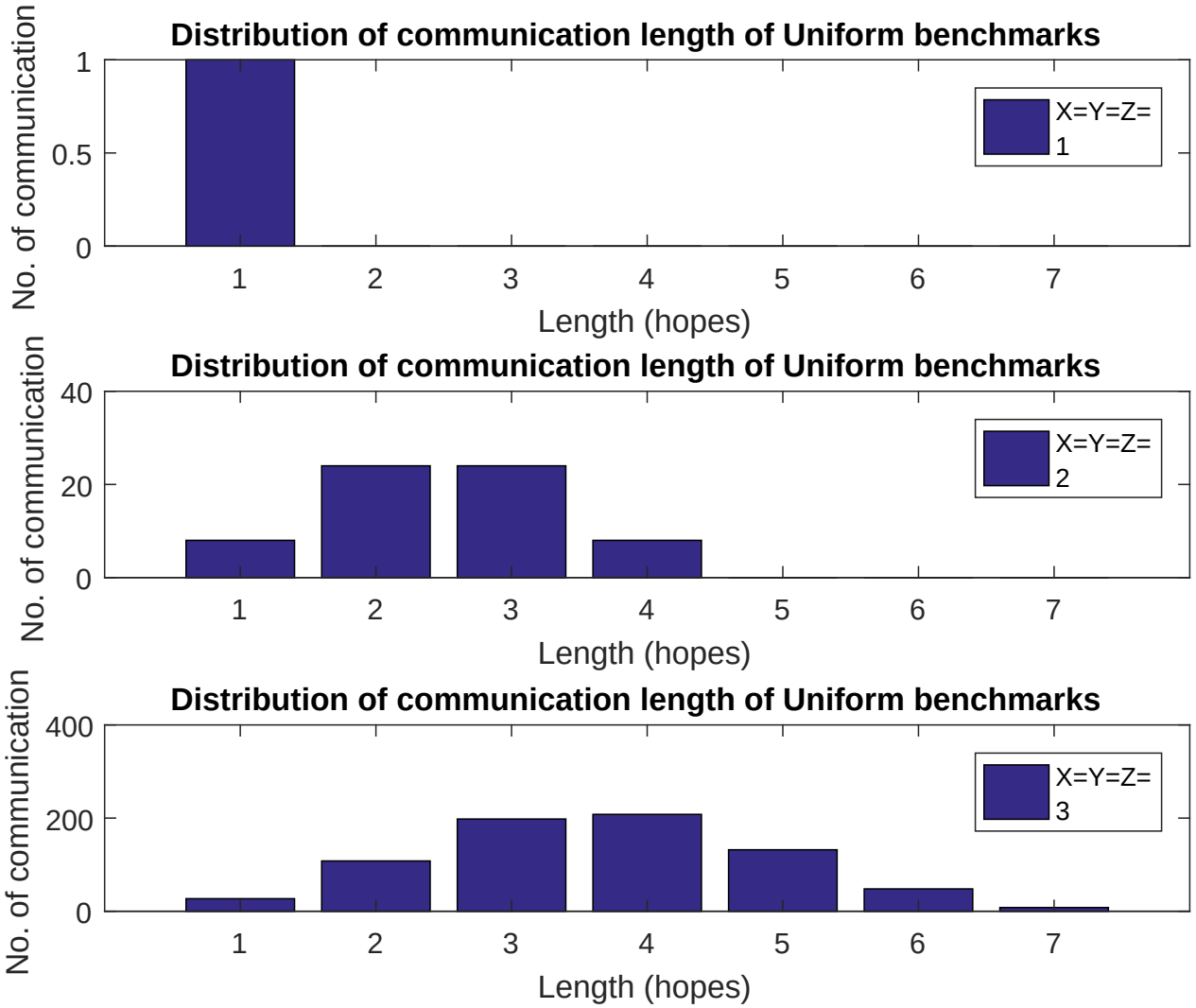
Therefore, probability of a communication is:

$$P_{S==(a,b,c) \text{ and } D==(m,n,p)} = 1/(X \times Y \times Z)^2 \tag{3}$$

Note: if (a,b,c) is eliminated from the destination list, $P_{D==(m,n,p)} = 1/(X \times Y \times Z - 1)$ and $P_{S==(a,b,c) \text{ and } D==(m,n,p)} = 1/((X \times Y \times Z)(X \times Y \times Z - 1)$

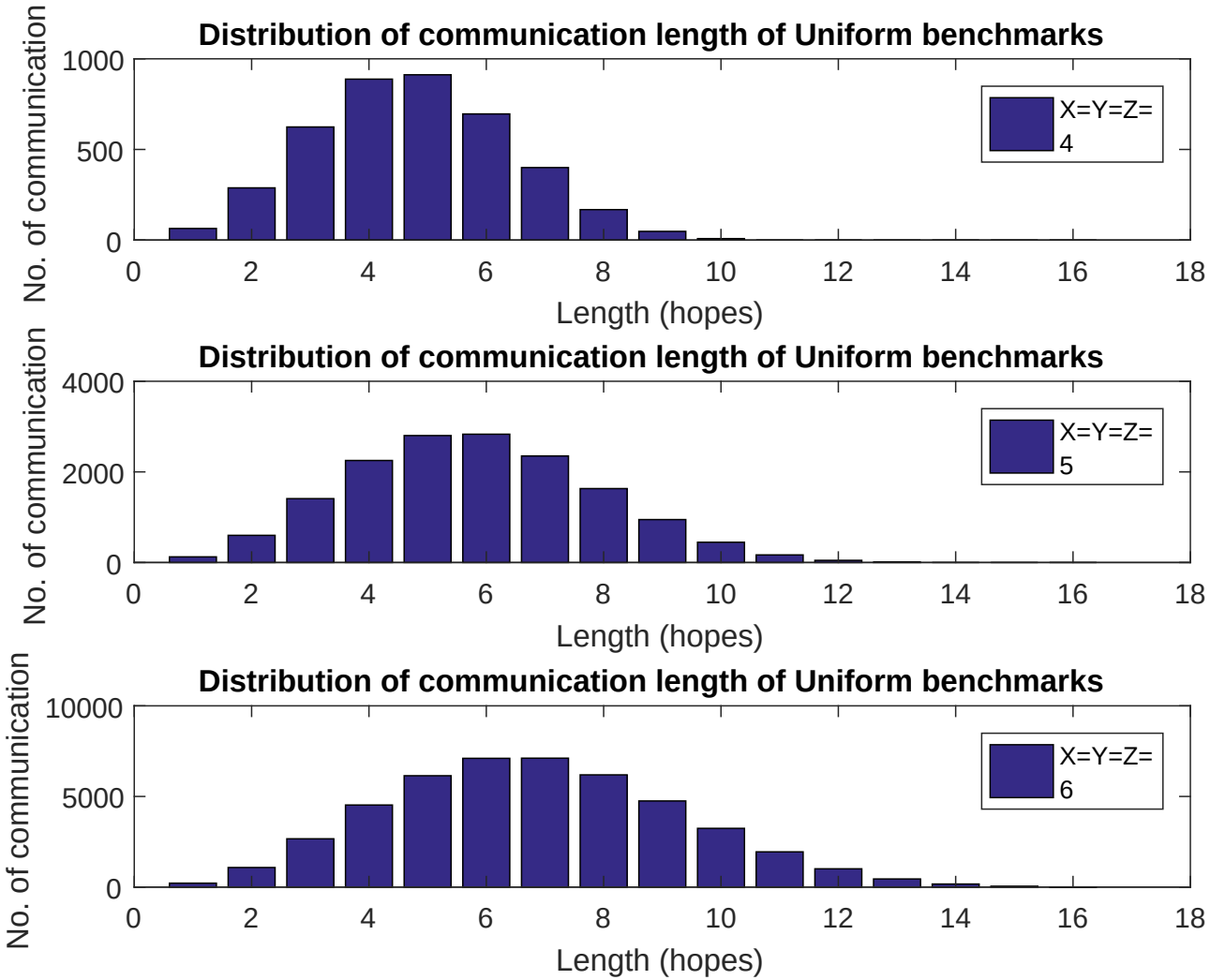
Uniform Distribution: Communication Length

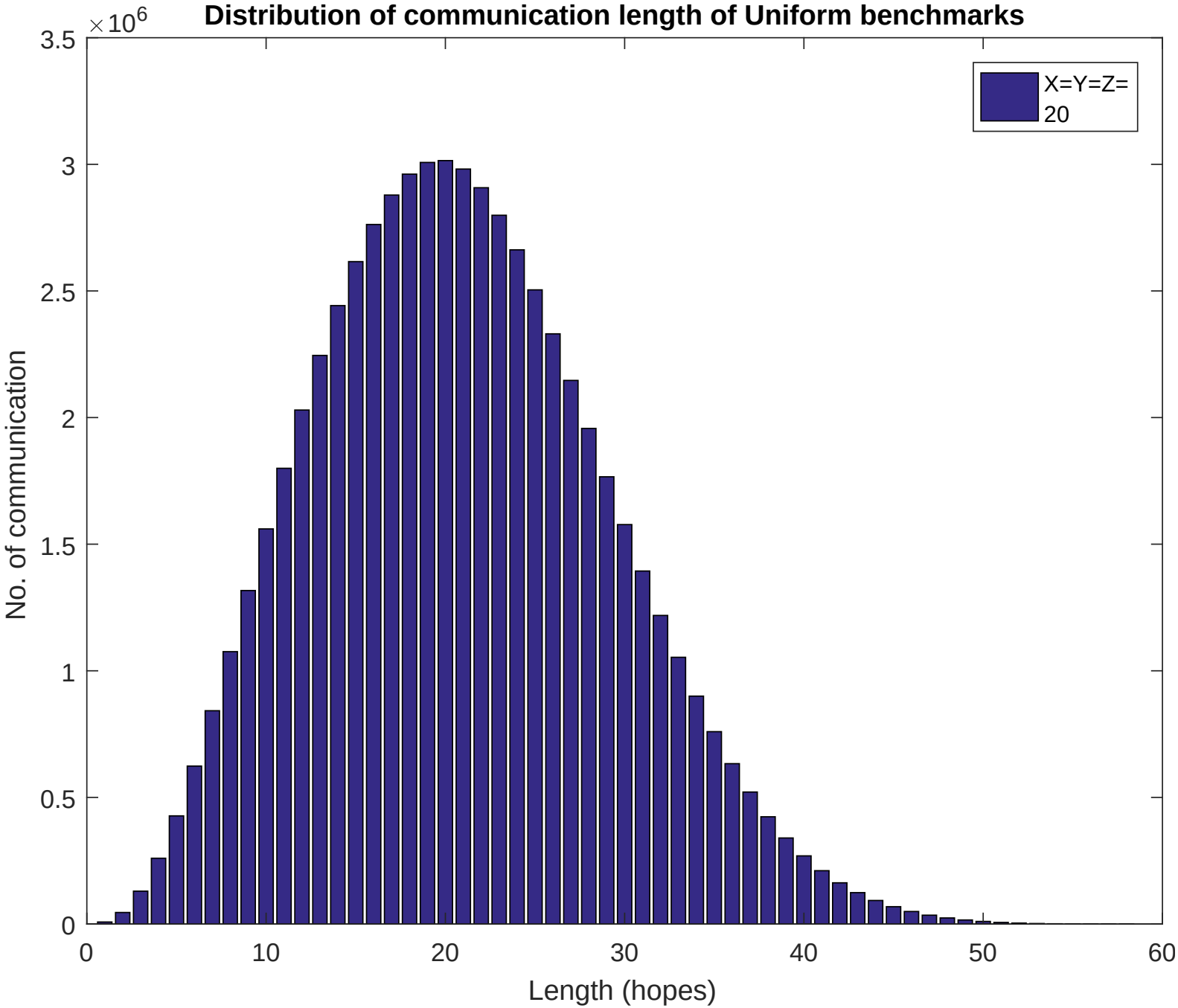
▸ Matlab code



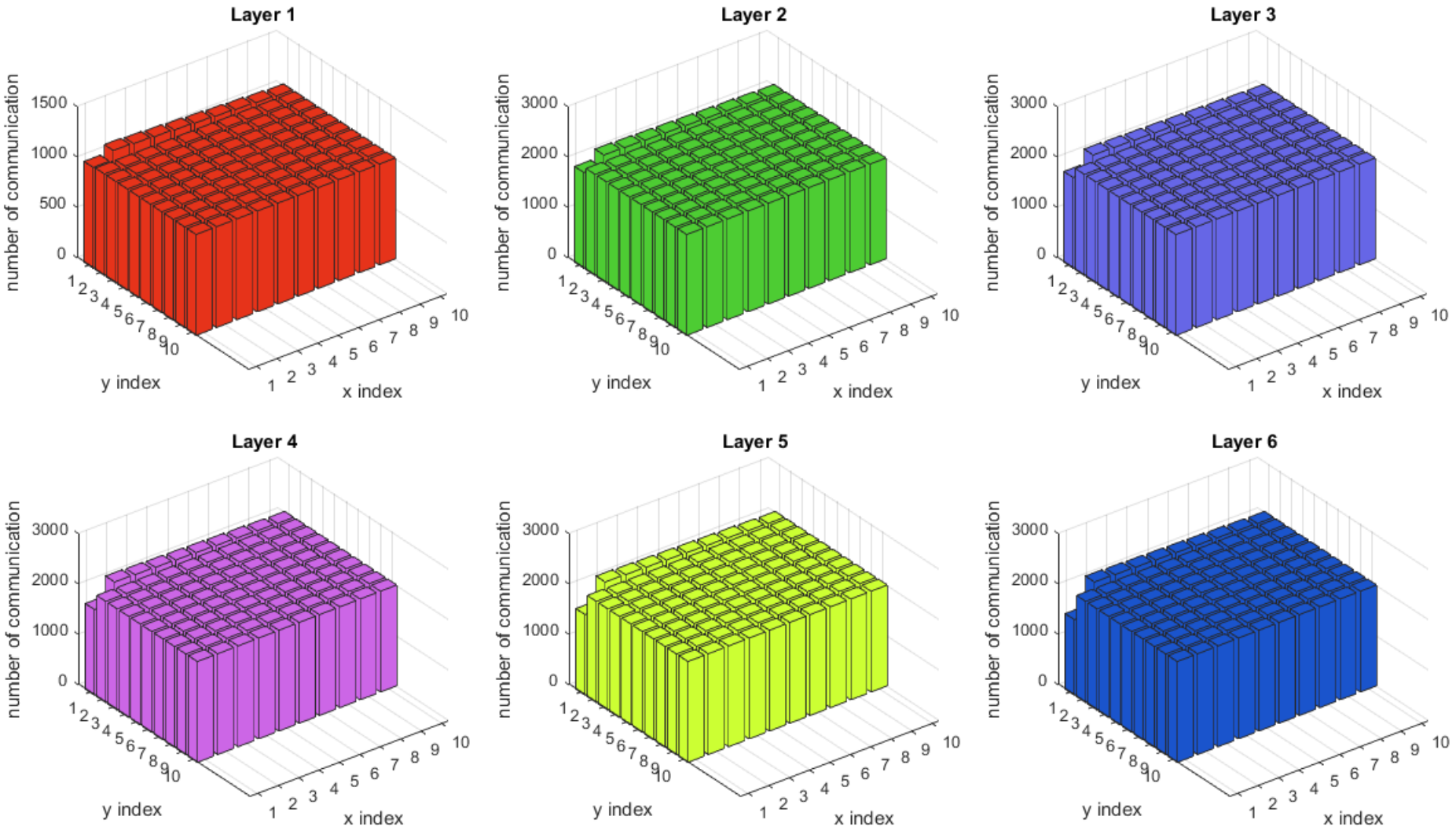
Uniform Distribution: Communication Length (2)

▸ Matlab code




Distribution of communication length of Uniform benchmarks
×10^6
3.5
3
2.5
2
1.5
1
0.5
0
No. of communication
X=Y=Z=
20
0
10
20
30
40
50
60
Length (hopes)

Uniform: Bottle-neck Analysis / network (10,10,10)



Note: This analysis is based on XYZ routing. Each bar = one router.

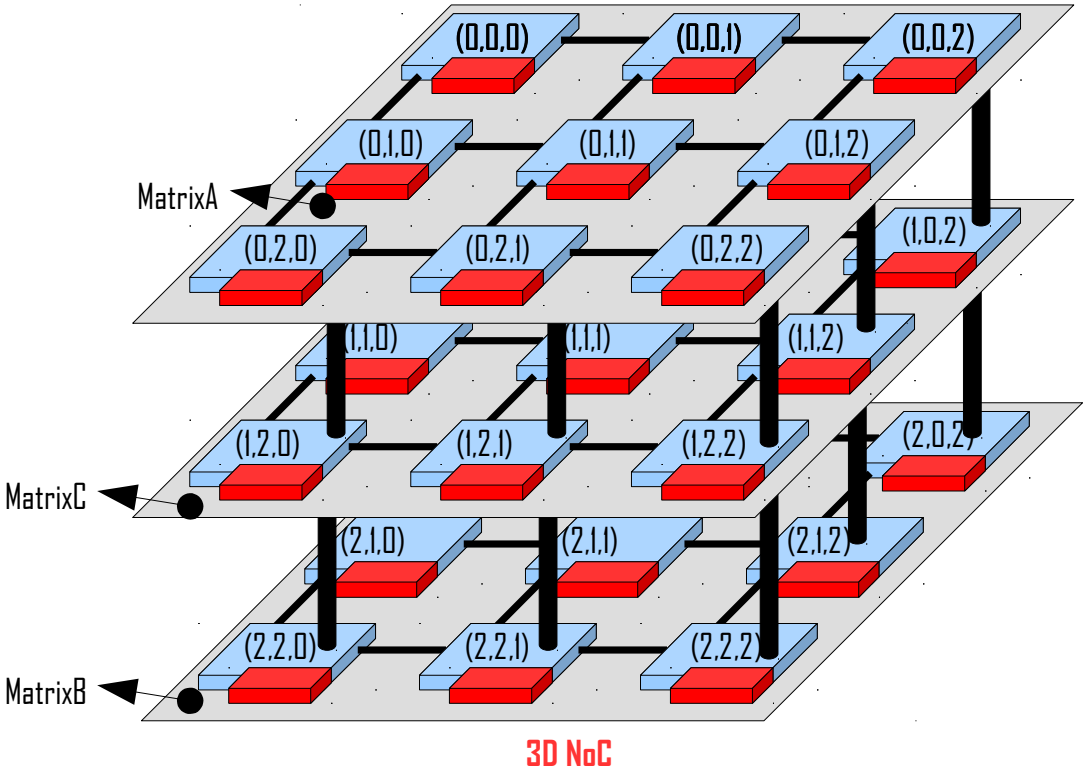
Uniform algorithm

Algorithm 2: Uniform Algorithm.

```
// Network
Input: Network(X, Y, Z)
// Amount of data for each communication
Input: D
// Communication set
Output: C = {c_i : (source → destination, amount of data)}
1 foreach node (a,b,c) in Network(X, Y, Z) do
2 |   foreach node (m,n,p) in Network(X, Y, Z) do
3 |   |   add ((a, b, c) → (m, n, p), D packets) to C
4 |   end
5 end
6 return C
```

Matrix-multiplication

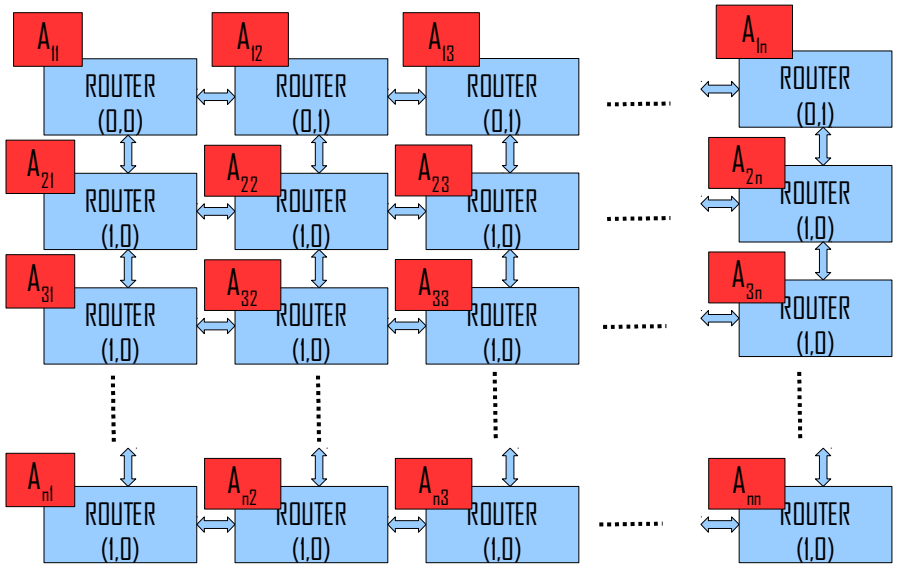
Matrix multiplication is one the most common functions for: scientific application, image/video processing, financial calculation. The calculation can be accelerated by utilizing the parallelism of NoC.
Let consider a multiplication: $C = A \times B$ where A,B,C has size (n,n). To calculate the function, matrix A, B, C are mapped to separated layers.



Matrix-multiplication: Mapping of matrix A

This is how matrix A is mapped. The similar idea is applied for matrix B and C.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \dots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & A_{n3} & \dots & A_{nn} \end{bmatrix}$$



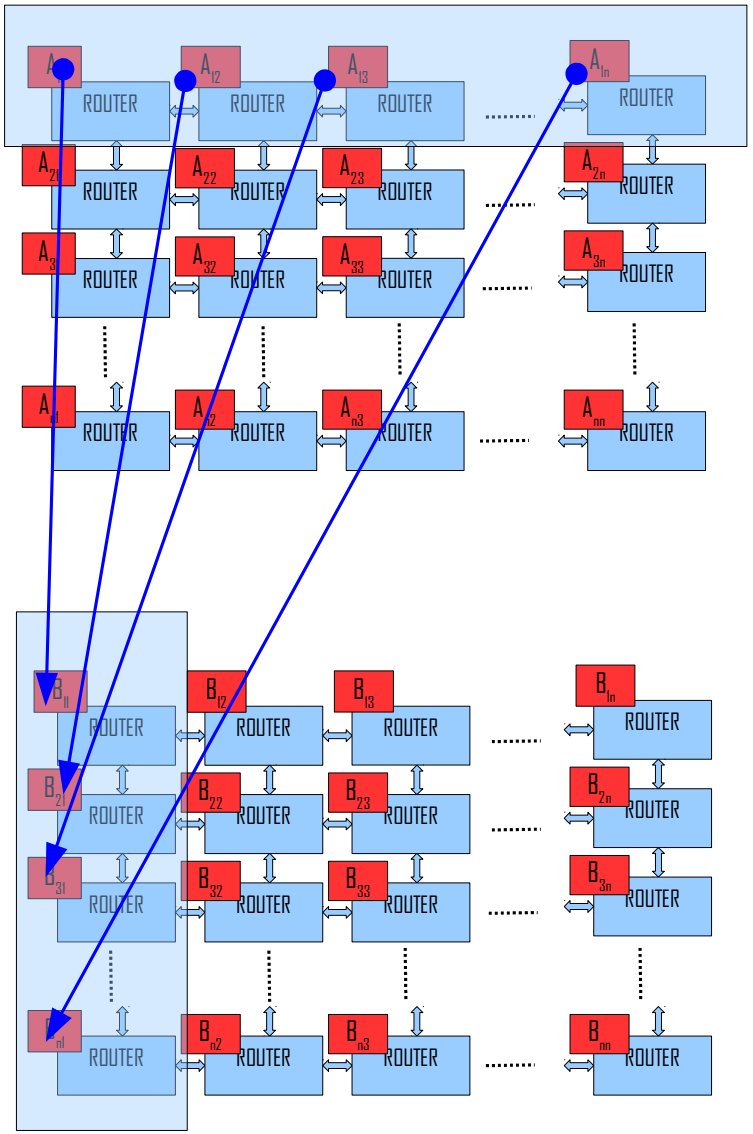
Matrix-multiplication: Sending data from A to B

Final equation:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj} \qquad (4)$$

Step 1: Element A_{ik} of matrix A is sent and **multiplied** to element B_{ji} of matrix B.

$$R_{ji} = A_{ij} \times B_{ji} \qquad (5)$$



Matrix-multiplication: Sending data from A to B

Step 2: Result R_{ij} is sent and **accumulated** to obtain matrix C.

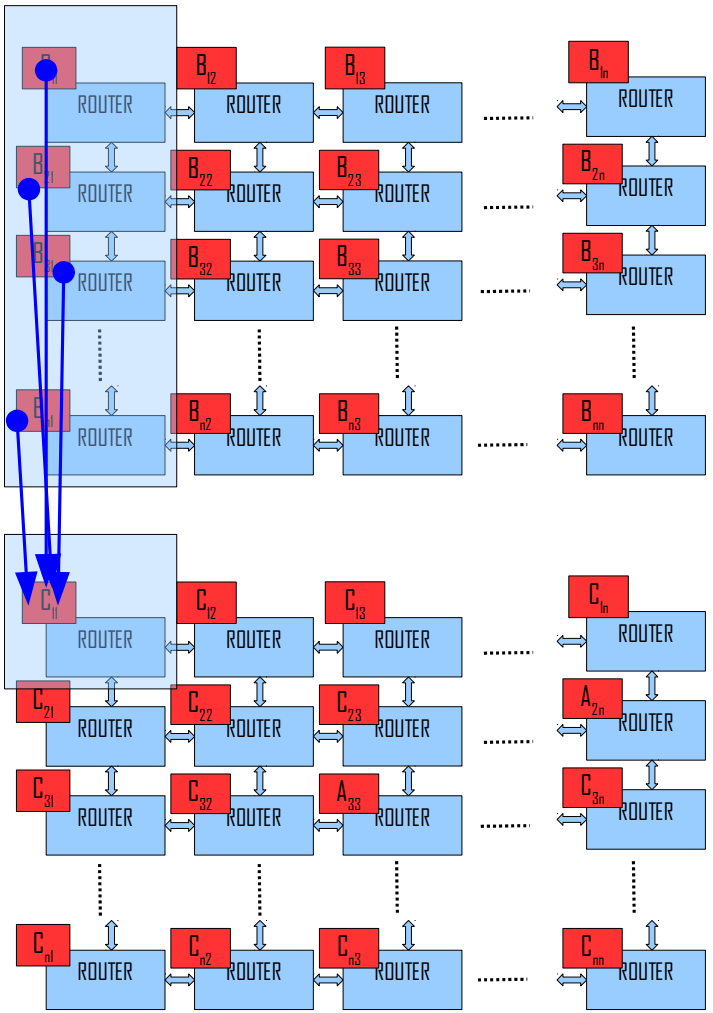
$$C_{ij} = \sum_{k=1}^{n} R_{ki} \tag{6}$$

Where:

$$R_{ji} = A_{ij} \times B_{ji} \tag{7}$$

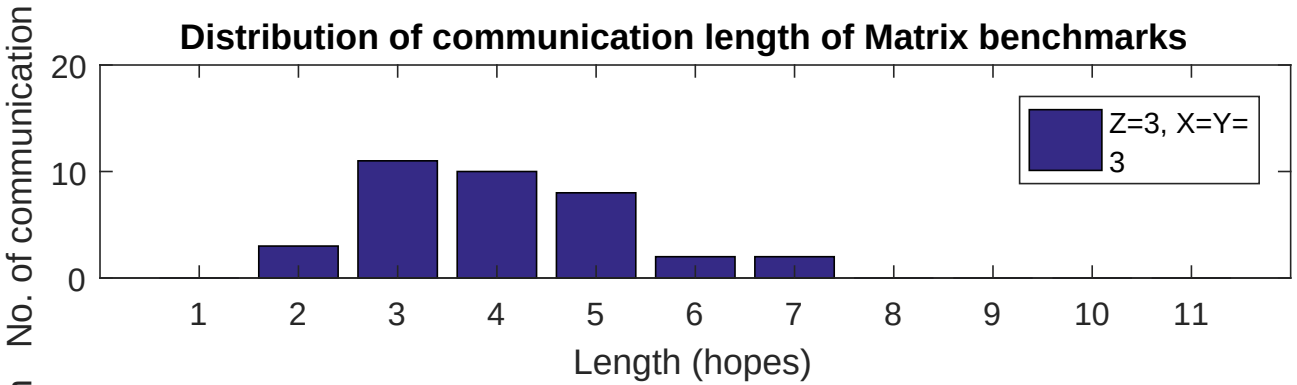
Therefore:

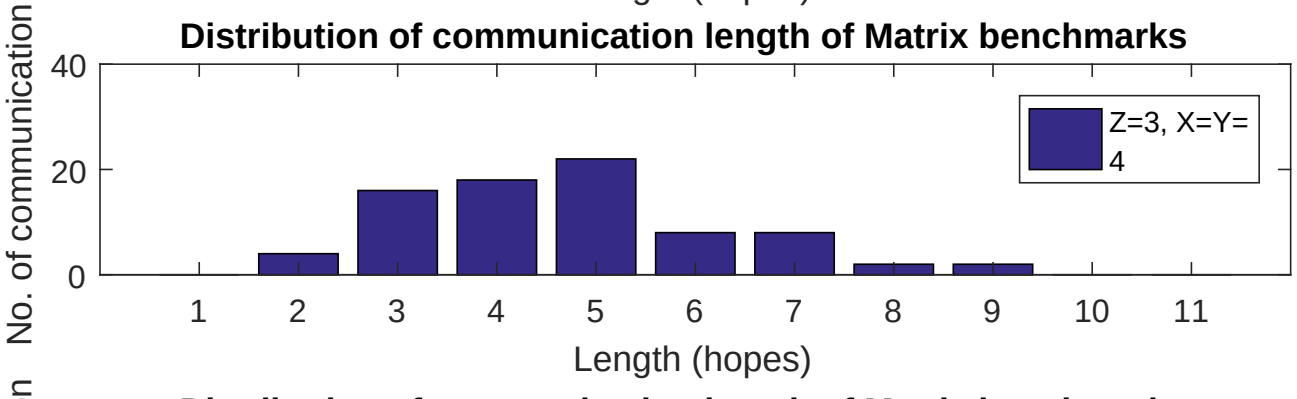
$$C_{ij} = \sum_{k=1}^{n} A_{ik} \times B_{kj} \tag{8}$$

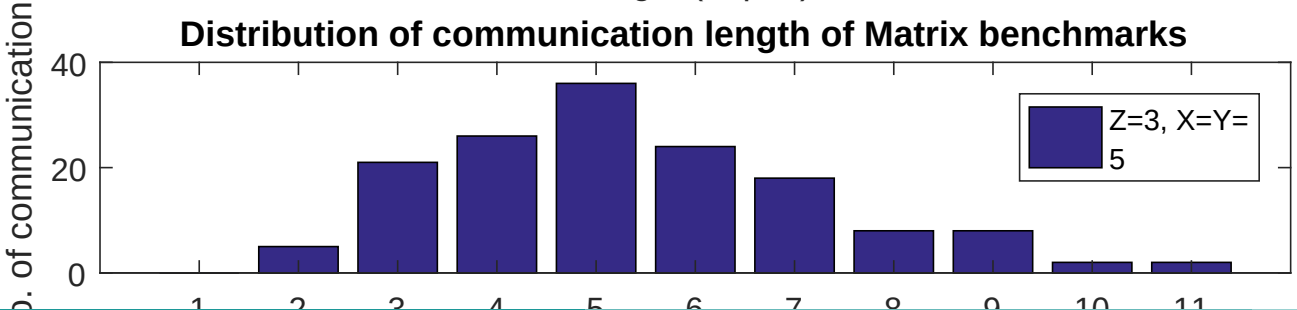


Matrix-Multiplication Distribution: Communication Length

▸ Matlab code

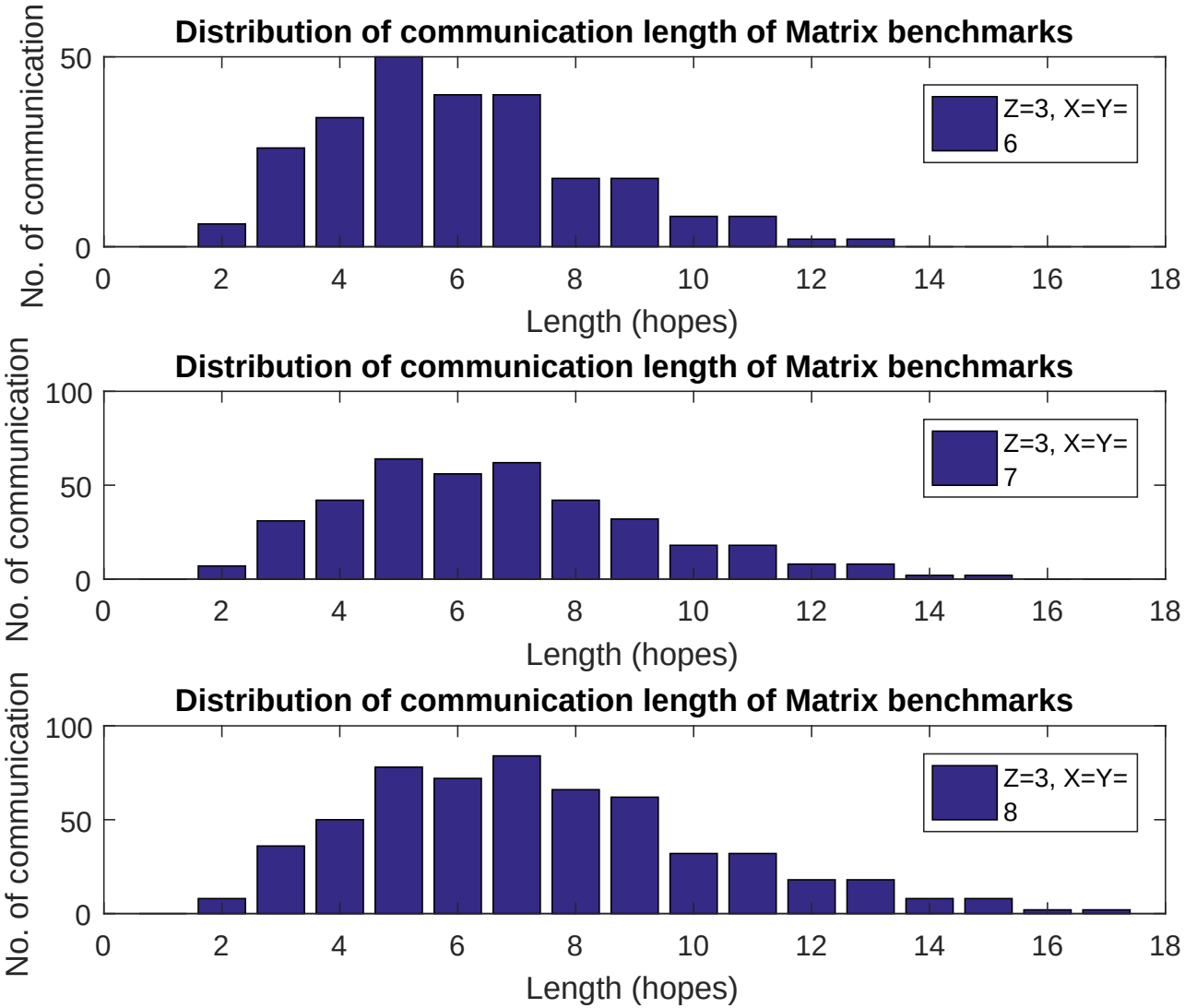


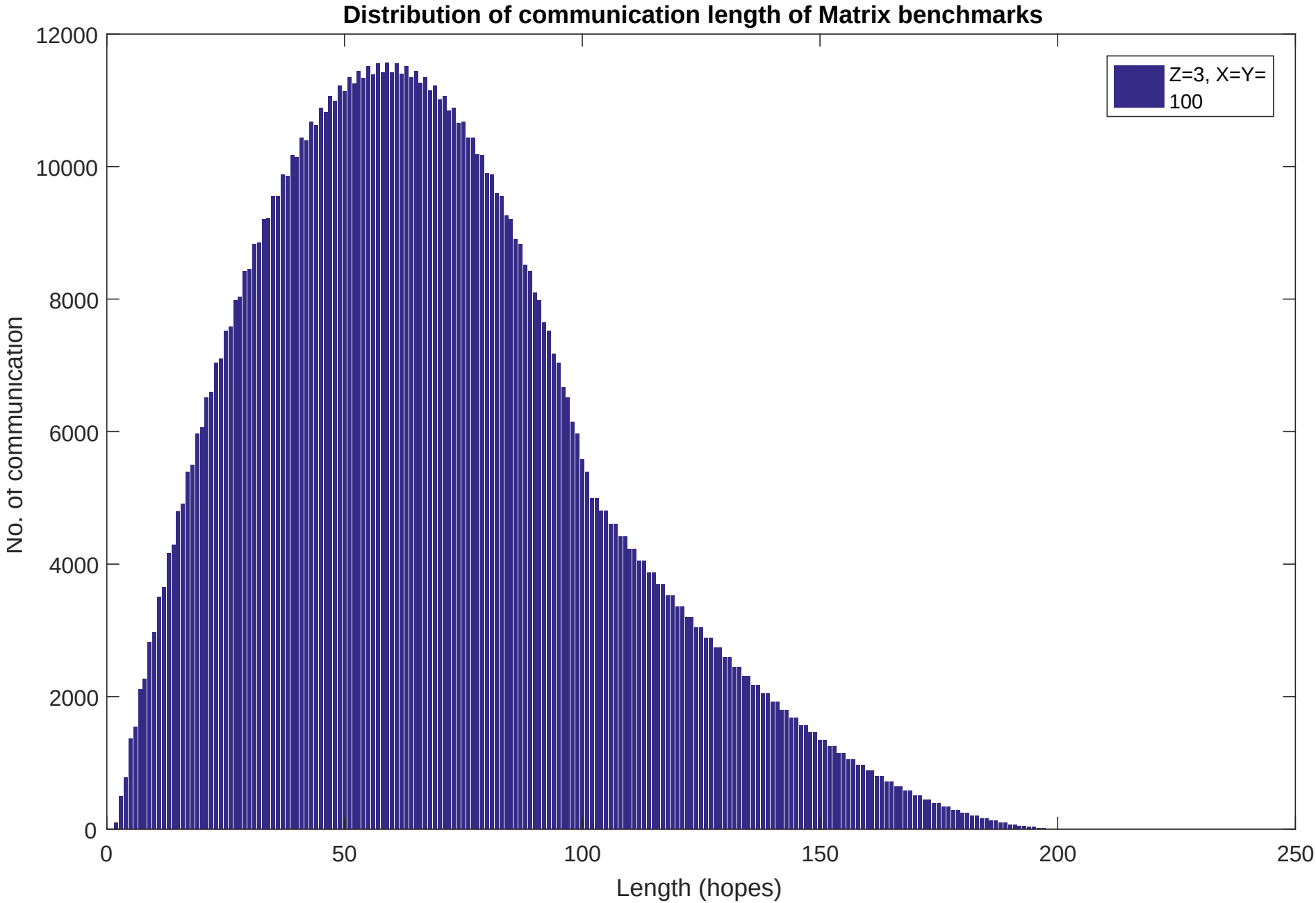




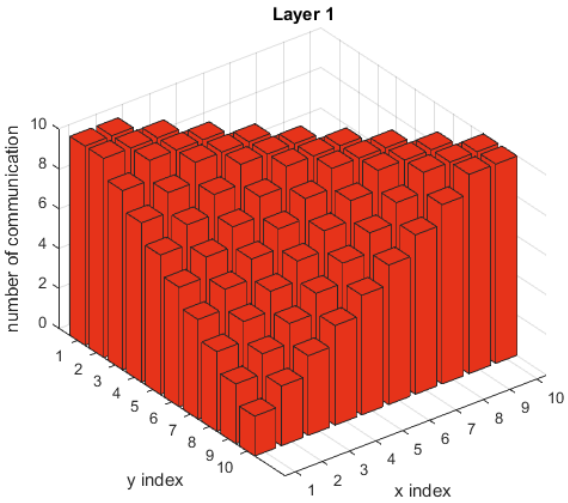
Matrix-Multiplication Distribution: Comm. Length (2)

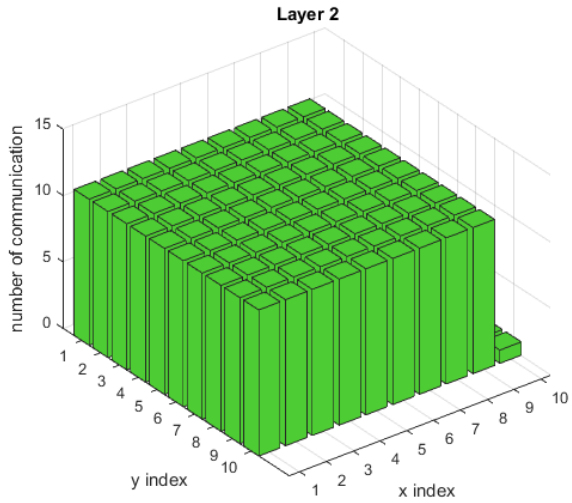
▸ Matlab code

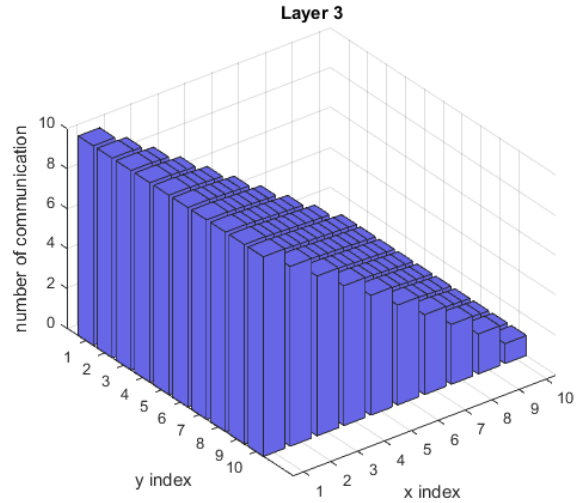


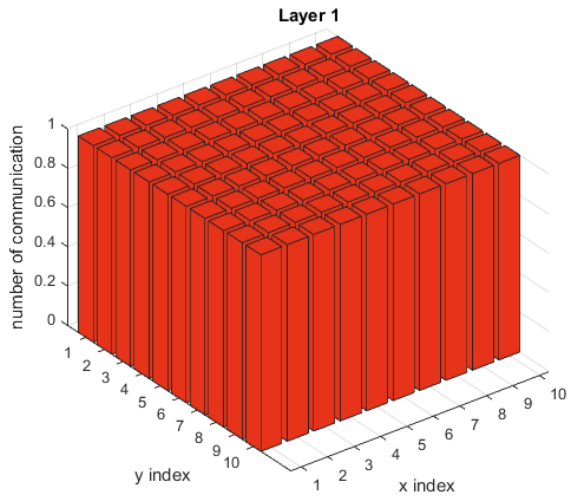

Distribution of communication length of Matrix benchmarks
Z=3, X=Y=100
No. of communication
Length (hopes)
0
2000
4000
6000
8000
10000
12000
0
50
100
150
200
250

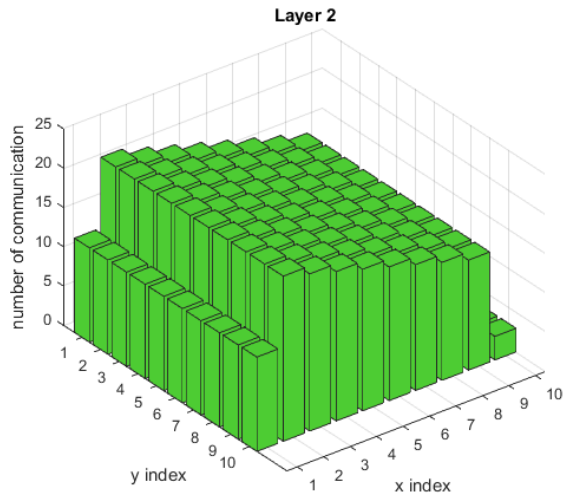
Matrix-multiplication: Bottle-neck Analysis / network (X=10,Y=10,Z=3)

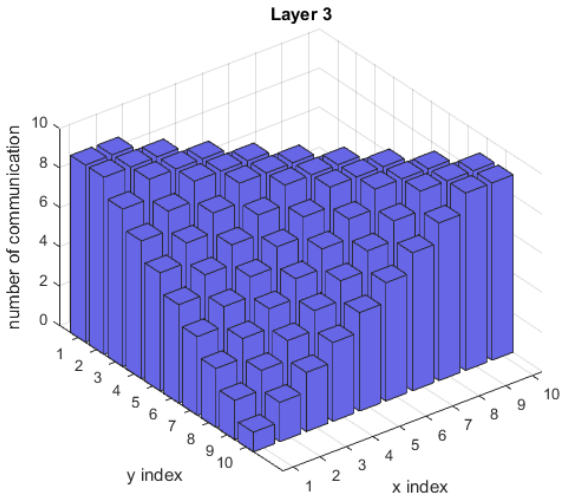












Matrix-multiplication algorithm

Algorithm 3: Matrix-multiplication Algorithm.

Input: *layerA*(n, n), *layerB*(n, n), *layerC*(n, n),
Input: $A(n, n)$, $B(n, n)$
Output: $C(n, n)$

```
1  foreach node (i,j) in layerA(n, n) do
2  |   send A(i,j) → layerB(j,i)
3  end
4  foreach node (i,j) in layerB(n, n) do
5  |   receive A(j,i)
6  |   R(i, j) = A(j, i) × B(i, j)
7  |   foreach k in 1:n do
8  |   |   send R(i,j) → layerC(i,k)
9  |   end
10 end
11 foreach node (i,j) in layerC(n, n) do
12 |   foreach k in 1:n do
13 |   |   send C(i,j) = C(i,j) + R(k,i)
14 |   end
15 end
16 return C(n, n) from layerC(n, n)
```

Hotspot 10%

Hotspot is similar to **Uniform**. Each nodes send messages to all non-hotspot node with an equal probability. The hotspot nodes are received extra E% of probability (e.g: 10%). For example: a network with size (3,3,3). A node (a,b,c) sends the same amount of packets to all node (0,0,0) $\rightarrow$ (2,2,2). In a network (X,Y,Z), probability of source node (a,b,c) is:

$$P_{S==(a,b,c)} = 1/(X \times Y \times Z) \tag{9}$$

Hotspot algorithm

Algorithm 4: Hotspot Algorithm.

```
// Network
Input: Network(X, Y, Z)
// Amount of data for each communication
Input: D
// Extra percentage of hotspot node
Input: E
// Communication set
Output: C = {c_i : (source → destination, amount of data)}
foreach node (a,b,c) in Network(X, Y, Z) do
    foreach node (m,n,p) in Network(X, Y, Z) do
        if node (m,n,p) is hotspot node then
            add ((a, b, c) → (m, n, p), (D+D*E/100) packets) to C
        else
            add ((a, b, c) → (m, n, p), D packets) to C
        end
    end
end
return C
```

Realistic Traffic Pattern

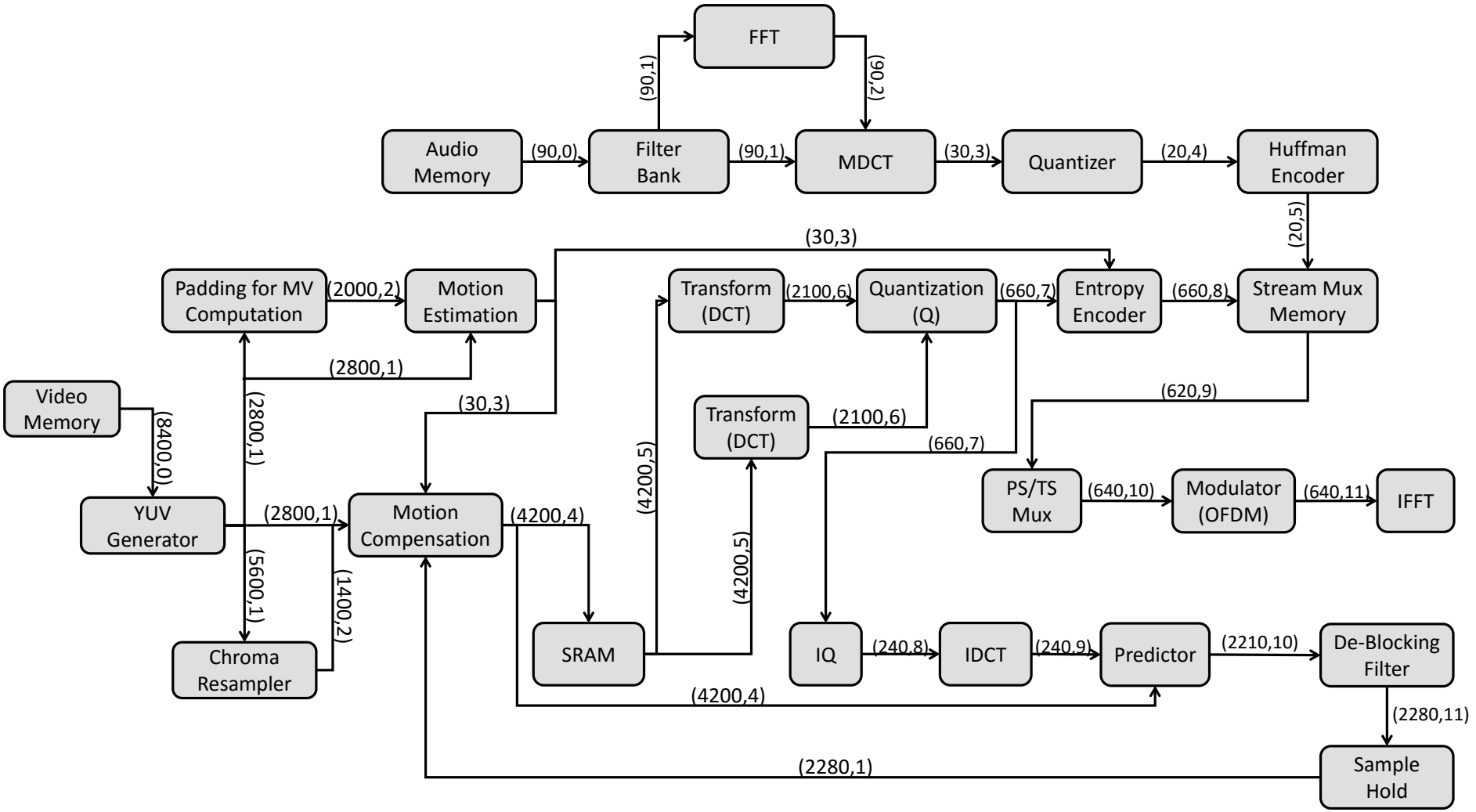
Realisc Traffic Pattern is design to import a mapped task graph to perform a simulation. It requires:

- A task graph from an application.
- Post-mapped topology from the task graph.
- Information of communication (source $\rightarrow$ destination, amount of data).
- Network on Chip model.

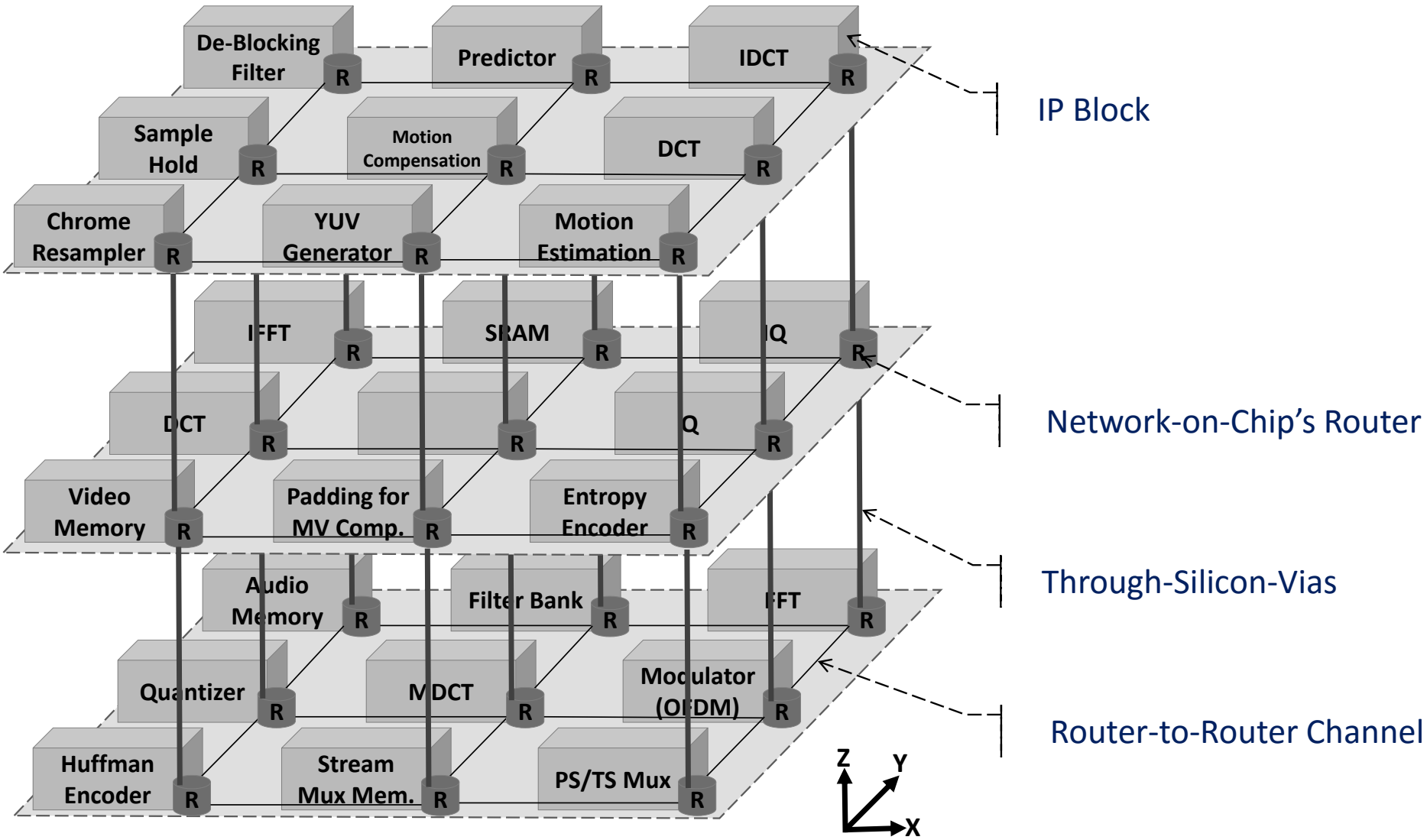
Source code (Verilog HDL): ▸ link

An example task graph: H.264 Encoder

A communication has an extra number: (D, O). D: amount of data (packets), O: order of communication.



An example of mapped application to 3D NoC



Algorithm of Realistic Benchmark

Algorithm 5: Realistic Benchmark Algorithm.

Input: $Network(X, Y, Z)$
// Communication set
Input: $C = \{c_i : (source \rightarrow destination, D, O)\}$

```
1  ProgramCounter = 0;
2  foreach node (i,j,k) in Network(X, Y, Z) do
3  |   foreach c_i in C do
4  |   |   if c_i(source) == (i, j, k) and ProgramCounter == O then
5  |   |   |   send (i,j,k) → c_i(destination) with c_i(D) packets.
6  |   |   end
7  |   |   if c_i(destination) == (i, j, k) and ProgramCounter == O then
8  |   |   |   receive c_i(D) packets.
9  |   |   end
10 |   end
11 end
12 if all destinations completedly receive their own c_i(D) packets then
13 |   ProgramCounter++;
14 end
```

Note: The analysis for these benchmarks are not completed.

Conclusion

- This presentation shows the details of how the benchmarks work.
- I explain the algorithm and the distribution of the benchmarks:
  - Transpose
  - Uniform
  - Matrix-multiplication
  - Hotspot 10%
  - Realistic Traffic Pattern
- The content can be used for further research.
- The whole analysis code can be download from: this repository.

Current & future works:

Task	Deadline	Note
TSV fault-tolerance	2017-Jan-31	Design, simulation and comparison
Paper writing	2017-Feb-28	Several papers

References I

[1] A. A. Chien and J. H. Kim, "Planar-adaptive routing: low-cost adaptive networks for multiprocessors," *Journal of the ACM (JACM)*, vol. 42, no. 1, pp. 91–123, 1995.

[2] R. Sivaram, "Queuing delays for uniform and nonuniform traffic patterns in a MIN," *ACM SIGSIM Simulation Digest*, vol. 22, no. 1, pp. 17–27, 1992.

[3] P. Chen, K. Dai, D. Wu, J. Rao, and X. Zou, "The parallel algorithm implementation of matrix multiplication based on ESCA," in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 1091–1094, IEEE, 2010.

[4] A. S. Zekri and S. G. Sedukhin, "The general matrix multiply-add operation on 2D torus," in *20th International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 8–16, IEEE, 2006.

References II

[5] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*.
Elsevier, 2004.

[6] A.-M. Rahmani, K. R. Vaddina, K. Latif, P. Liljeberg, J. Plosila, and H. Tenhunen, "High-performance and fault-tolerant 3D noc-bus hybrid architecture using arb-net-based adaptive monitoring platform," *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 734–747, 2014.

[7] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113–129, 2005.

Thank you for your attention!