# Communication Concerns

## 1. Topology

<table>
<thead>
<tr>
<th>Topology</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline</td>
<td>Matrix Multiplication, Backward-Substitution, Insertion Sort, Adding Number, Sieve of Eratosthenes</td>
</tr>
<tr>
<td>Star</td>
<td>Pixel-by-Pixel Image Processing, Monte-Carlo Simulations,</td>
</tr>
<tr>
<td>Mesh</td>
<td>Finite Difference Method, Matrix Multiplication, Jacobi Iteration, Cellular Automata</td>
</tr>
<tr>
<td>Full Mesh</td>
<td>LU Factorization.</td>
</tr>
<tr>
<td>Hypercubic</td>
<td>Hypercube Quicksort, Hypercube Mergesort, Reduction, Broadcast and Multicast, Find Maximum</td>
</tr>
<tr>
<td>Butterfly</td>
<td>FFT, Prefix Sum</td>
</tr>
<tr>
<td>Tree</td>
<td>Mergesort, Barnes-Hutt, Bucket Sort, Adaptive Quadrature Numerical Integration</td>
</tr>
<tr>
<td>Transpositional</td>
<td>Transpositional FFT, Shear Sort</td>
</tr>
<tr>
<td></td>
<td><em>Special case of mesh, butterfly</em></td>
</tr>
</tbody>
</table>
| Odd-Even | Gauss-Seidel Method, Odd-Even Insertion Sort  
| Special case of pipeline. |
| Irregular | Cholesky Factorization, Sparse matrix operations,  
| Event-driven, Discrete Event Simulations,  
| Fock Matrix Computations. |
| Multigrid solver |

* some topology is more primitive than others.

### 2. Determination

a) Topology determined statically.

Communication pattern is well defined from the start, fixed at some kind of a topology. The set of nodes who will participate in the process (local, global) are predetermined.

b) Topology determined dynamically.

→ Topology stay fixed.

Program starts with speculative communication, depending on the outcome of the previous computations. After the dependent values are computed, communication topology is determined, and it stays fixed throughout the life cycle of execution.

→ Topology changes in phases.

Determined dynamically, but periodically change during phases.

### 3. Frequency

a) Periodic in phases. (a.k.a. loosely synchronous)

There is periodic global communication step after each computational phase.

Until all other processes are done, a process cannot go into the communication step.
This is enforced strictly in each of the phases, thus each process will start the subsequent computational phases with exactly same updated values.

b) Relaxed Periodic
For processes, some relaxation of the constraint is introduced to enter the global communication phase or the next phase early, so that it does not have to wait for all other processes. However, periodicity of the program is still maintained through some window size, so that one process will not go too far out of sync from other processes.

c) Chaotic Periodic
The phases still exists, but a process is almost careless on where other processes stand in the computation. No window size or any constraint is enforced, thus a process does not wait for other processes. This can only be used if convergence is still guaranteed, or dependency relationship automatically enforces order of computation.

4. Volume
a) Logarithmic change in subsequent communication
As the phase advances in the program, a node is responsible for communicating logarithmically increasing/decreasing volume of data (1, 2, 4, 8 …). In many cases received data is processed in concatenation. (Prevalent in divide-and-conquer algorithms Mergesort, Quicksort)

b) Additively incremental change
Subsequent phases require additively incremental volume to be communicated. (Prevalent in pipeline algorithms)

c) Pairwise / One-To-One at fixed volume
Communication volume is always fixed at certain amount throughout the life cycle of the program. Previous received data is accumulated rather than concatenated in the next communication volume. (adding number in pipeline, prefix sum)

d) Aggregated amount
Volume is aggregated, and does not portray any specific pattern.
e) Scattered amount
Large amount of data is scattered to individual pairwise data amount.

5. Initiation

a) Sender-Initiated: Definer knows user
   Sender knows the exact identity (i.e. node number) of the receiving process.

b) Receiver-Initiated: User knows definer
   Receiver requests data to the sender. Sender, upon blindly listening for any request,
   responds by sending desired data.

c) Neither knows
   Neither side knows who is receiving what. (* mechanism).

d) Both (Collective)
   Sender and receiver both participates in a collective call. (i.e. broadcast, multicast,
   etc)
   * discuss in relation to library calls and programming primitives.
   (send/recv, put/get, assignment)