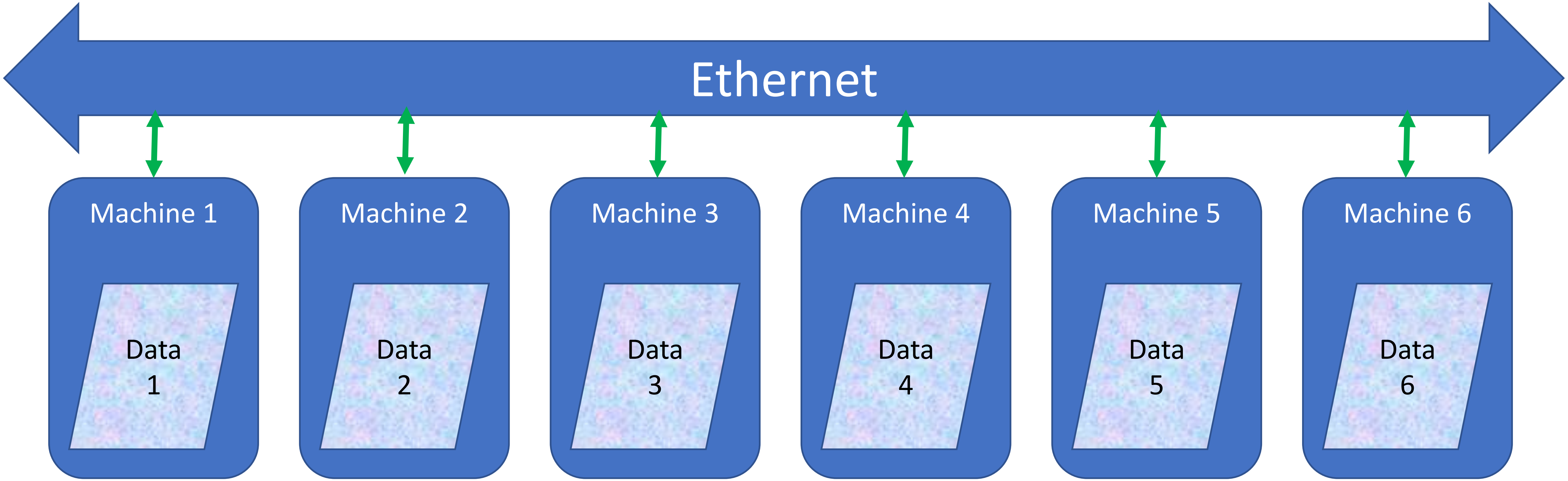
Distributed Sorting

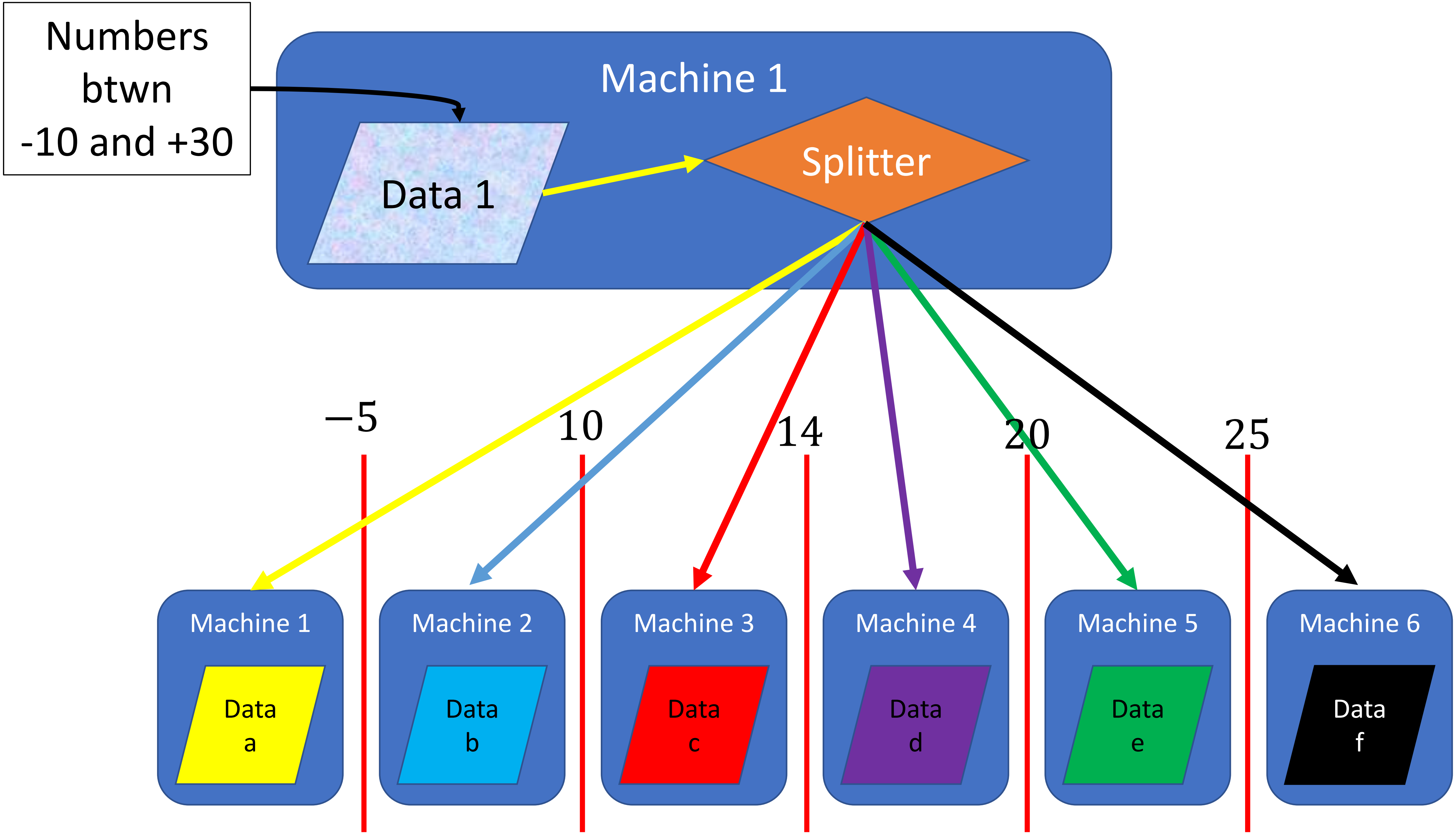
Sorting Basics

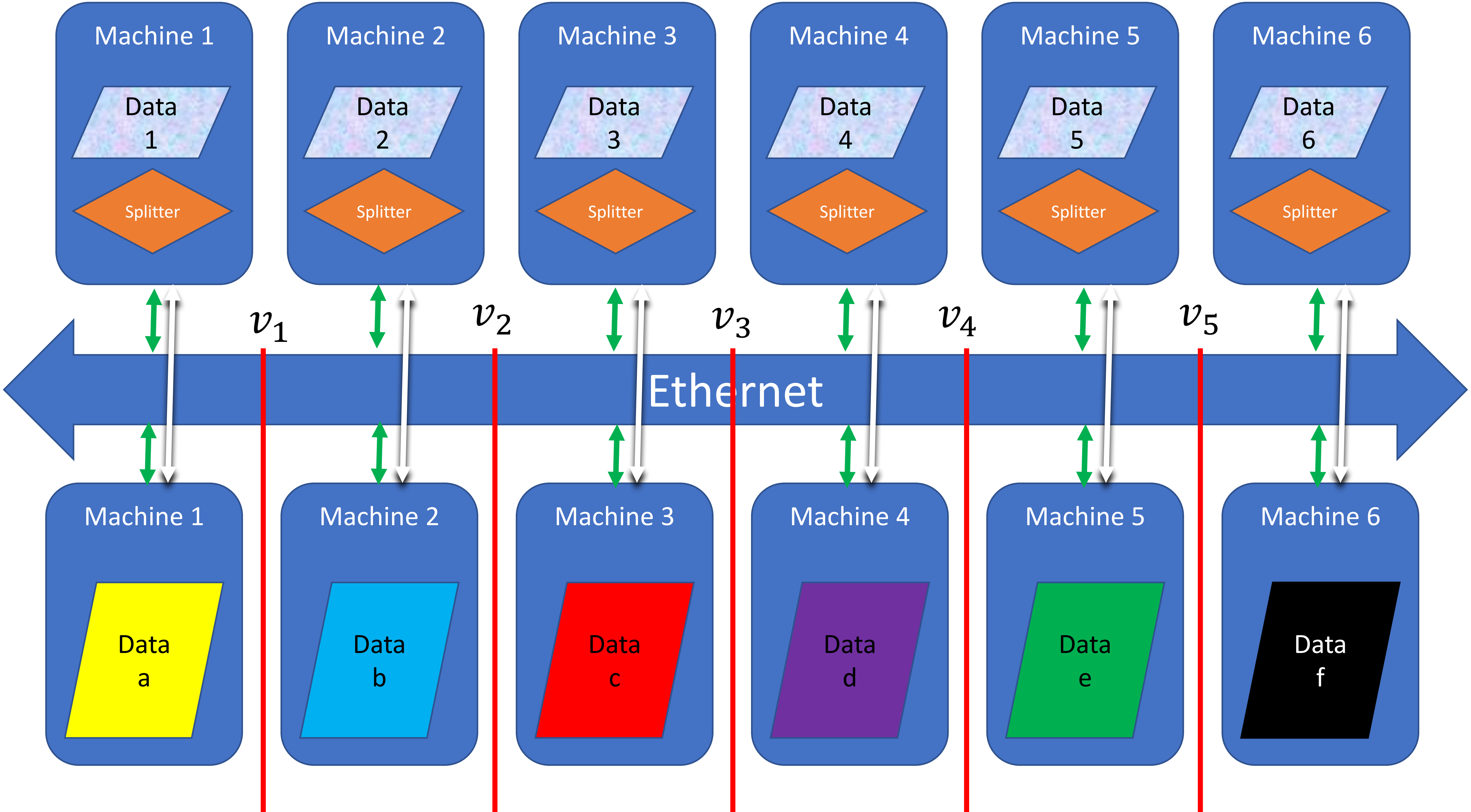
- Task: sort n elements in increasing or decreasing order
- Elements can be numbers, strings, keys…
- Bubble sort: comparing neighboring entries: $O(n^2)$ time
- Quick-Sort Merge-Sort: $O(n \log n)$ time
- In general: $\Omega(n \log n)$ time is the best possible.
- Bucket-sort: when distribution of the elements is known, it is possible to sort in $O(n)$ time

Distributed sorting

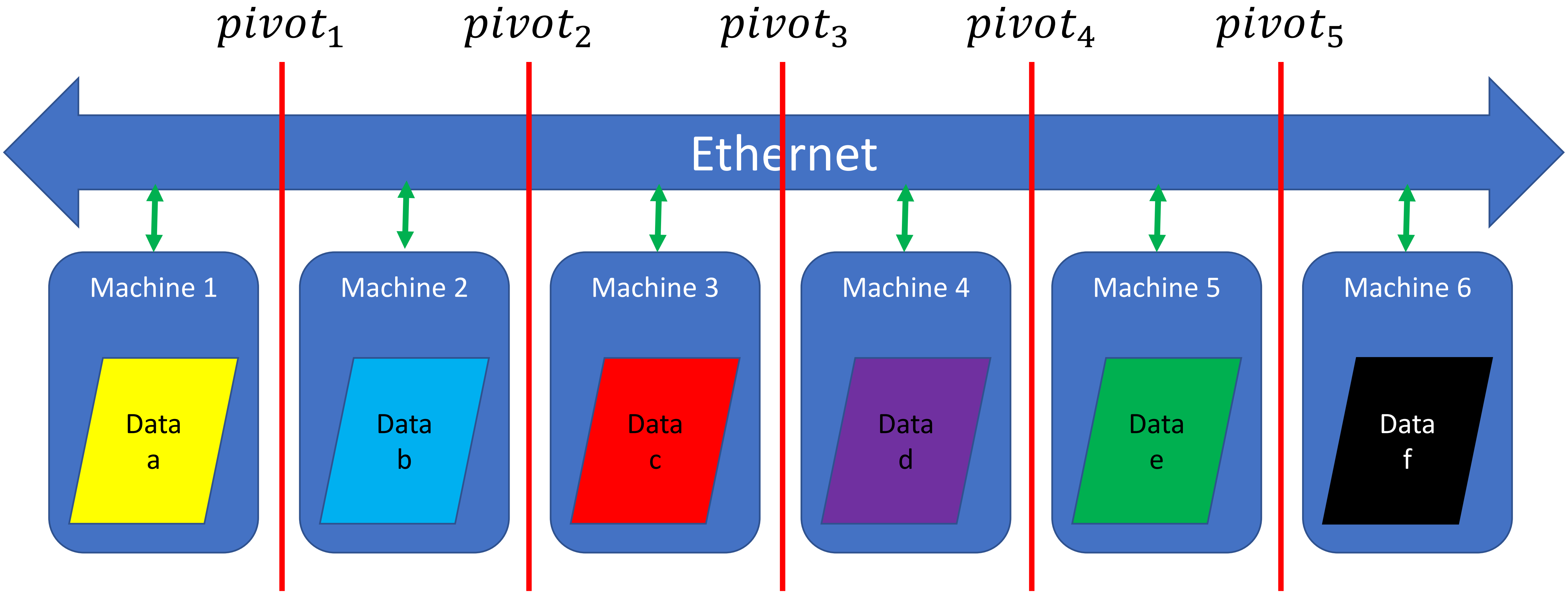
- Suppose we want to sort a file of 60 GB distributed across 6 machines.
- The main bottleneck is the communication between the machines.




Numbers btwn -10 and +30
Machine 1
Data 1
Splitter
-5
10
14
20
25
Machine 1
Data a
Machine 2
Data b
Machine 3
Data c
Machine 4
Data d
Machine 5
Data e
Machine 6
Data f


Machine 1
Data 1
Splitter
Machine 2
Data 2
Splitter
Machine 3
Data 3
Splitter
Machine 4
Data 4
Splitter
Machine 5
Data 5
Splitter
Machine 6
Data 6
Splitter
v_1
v_2
v_3
v_4
v_5
Ethernet
Machine 1
Data a
Machine 2
Data b
Machine 3
Data c
Machine 4
Data d
Machine 5
Data e
Machine 6
Data f

- At the end , all elements in data a are smaller than all elements in data b, etc.
- Pivot points: $Data_a \le v_1 < Data_b \le v_2 < \mathrm{Data_c} \le \cdots$



Finding good pivots

- Goal: choose $v_1, v_2, \ldots, v_5$ so that each of the 6 computers receives a similar number of points.
- Restriction: Communicate a tiny fraction of the examples between computers.
- Idea: Use sampling.

Pivot selection algorithm

Distributed Algorithm (Assuming 6 computers, 5 pivots, k samples from each computer.)

1. Each computer selects k examples uniformly at random
2. Aggregator Computer collects all $6k$ examples.
3. Aggregator sorts examples $x_1 \leq x_2 \leq \cdots \leq x_{6k}$
4. Aggregator chooses pivots to be $x_k \leq x_{2k} \leq x_{3k} \leq x_{4k} \leq x_{5k}$
5. Pivots distributed to all machines.

Clearly splits sample to 6 equal parts

Using Probability Theory: split all data to approximately equal parts even if the samples are small.