**1. Introduction to Communication Operations**

**Core Concepts**

* **Communication Need**: Parallel algorithms require data exchange between processes, which introduces **interaction delays** impacting efficiency.
* **Patterns**:
  + **Global Operations**: All processes participate (e.g., broadcast).
  + **Local Operations**: Subsets of processes interact (e.g., neighbor exchanges).
* **Key Challenge**: Efficient implementation across architectures (e.g., hypercube, mesh).

**2. One-to-All Broadcast & All-to-One Reduction**

**Definitions**

* **One-to-All Broadcast**: Source process sends identical data (size m) to all others.
* **All-to-One Reduction**: Data from all processes combined (e.g., sum, max) at a single destination using an **associative operator**.

**Algorithms**

**Ring/Linear Array**

* **Naive Approach**: Sequential sends (bottleneck at source).
* **Optimized (Recursive Doubling)**:
  + **Steps**:
    1. Source sends to farthest node.
    2. Each step halves the distance (avoids congestion).
  + **Time**: log⁡plog*p* steps.
  + **Example**: 8-node ring (Figure 4.1):
    1. Step 1: Node 0 → Node 4.
    2. Step 2: Nodes 0,4 → Nodes 2,6.
    3. Step 3: Nodes 0,2,4,6 → Nodes 1,3,5,7.

**Mesh**

* **2D Implementation**:
  + **Phase 1**: Broadcast along rows.
  + **Phase 2**: Broadcast along columns.
  + **Time**: O(p)*O*(*p*​) for a p×p*p*​×*p*​ mesh.

**Hypercube**

* **Dimension-wise Communication**:
  + **Steps**: Logically traverse hypercube dimensions (MSB to LSB).
  + **Time**: log⁡plog*p* steps.
  + **Example**: 3D hypercube (Figure 4.2):
    - Step 1: Node 0 → Node 1 (dimension 2).
    - Step 2: Nodes 0,1 → Nodes 2,3 (dimension 1).
    - Step 3: Nodes 0-3 → Nodes 4-7 (dimension 0).

**Reduction**

* **Reverse of Broadcast**: Combine data hierarchically (e.g., sum in log⁡plog*p* steps).

**3. All-to-All Broadcast & Reduction**

**Definition**

* **All-to-All Broadcast**: Each process sends unique data to all others.
* **All-Reduce**: Combines data from all processes, results distributed to all.

**Algorithms**

**Ring/Linear Array**

* **Phases**:
  1. Each node sends consolidated data to neighbor.
  2. Forward data, extracting relevant parts.
* **Time**: p−1*p*−1 steps.
* **Example**: 8-node ring (Figure 4.3):
  1. Step 1: Nodes send all data left/right.
  2. Step 2: Nodes forward non-local data.

**Hypercube**

* **Dimension-wise Exchange**:
  + **Steps**: log⁡plog*p* steps, doubling data size each step.
  + **All-Reduce**: Replace concatenation with reduction (e.g., sum).

**4. Scatter & Gather**

**Definitions**

* **Scatter (One-to-All Personalized)**: Source sends unique data to each process.
* **Gather**: Reverse of scatter; unique data collected at one node.

**Hypercube Algorithm**

* **Scatter Steps** (Figure 4.4):
  1. Source sends half of messages to neighbor.
  2. Repeat recursively, halving data each step.
* **Time**: log⁡plog*p* steps.
* **Gather**: Reverse steps, concatenating data.

**5. All-to-All Personalized Communication (Total Exchange)**

**Definition**

* Each node sends distinct data (size m) to every other node.
* **Applications**: FFT, matrix transpose, sorting.

**Algorithms**

**Ring**

* **Phases** (Figure 4.5):
  1. Nodes send all data to neighbor.
  2. Extract local data, forward rest.
* **Time**: p−1*p*−1 steps.

**Mesh**

* **2D Phases**:
  1. **Row-wise**: Exchange clustered data.
  2. **Column-wise**: Complete total exchange.
* **Time**: O(p)*O*(*p*​).

**Hypercube**

* **Dimension-wise** (Figure 4.6):
  + **Steps**: log⁡plog*p*, exchanging p/2*p*/2 messages per step.
  + **Time**: m(p−1)*m*(*p*−1) for message size m.

**6. Circular Shift**

**Definition**

* **Permutation**: Node i sends data to node (i + q) mod p.
* **Applications**: Matrix rotations, pattern matching.

**Mesh Implementation** (Figure 4.7)

* **Steps**:
  1. **Row Shift**: Shift within rows.
  2. **Column Shift**: Adjust for wraparound.
* **Example**: 5-shift on 4×4 mesh:
  1. Phase 1: Row shifts (e.g., Node 0 → Node 1).
  2. Phase 2: Column shifts (e.g., Node 1 → Node 5).

**Key Takeaways**

1. **Broadcast/Reduction**: Fundamental for data distribution/aggregation.
2. **Scatter/Gather**: Efficient for personalized data exchange.
3. **All-to-All**: Critical for dense communication patterns (e.g., FFT).
4. **Topology Matters**: Hypercubes enable log⁡plog*p* steps; meshes use O(p)*O*(*p*​).
5. **Optimization**: Avoid congestion via recursive doubling or dimension-wise routing.

**Example Calculations**

1. **Hypercube Broadcast Time**:  
   For p=8*p*=8, log⁡28=3log2​8=3 steps.
2. **Ring All-to-All Time**:  
   p−1=7*p*−1=7 steps for 8 nodes.
3. **Mesh Shift Steps**:  
   5-shift on 4×4 mesh: 2 phases (row + column).

This structured breakdown ensures **no content is omitted**, with **clear definitions** of technical terms (e.g., associative operator, permutation) and **visualizable algorithms**. Let me know if you need further elaboration!

**1. What is the fundamental difference between one-to-all broadcast and one-to-all personalized communication (scatter)?**

**Answer**:

* **One-to-all broadcast**: Source sends the **same data** to all processes (1 message duplicated).
* **Scatter**: Source sends **unique data** to each process (p distinct messages).

**Example**:

* Broadcast: Sending a global configuration file to all nodes.
* Scatter: Distributing unique array chunks to different processes.

**2. Why is recursive doubling more efficient than sequential sending for broadcast on a ring?**

**Answer**:

* **Sequential**: Source sends p-1 messages sequentially → Bottleneck at source.
* **Recursive Doubling**:
  + Each step doubles the number of processes with data.
  + **Time**: log⁡plog*p* steps vs. p−1*p*−1 steps.
  + **Utilization**: All links used concurrently.

**Example**: On an 8-node ring, recursive doubling completes in 3 steps vs. 7 steps.

**3. How does all-to-all personalized communication differ from all-to-all broadcast?**

**Answer**:

* **All-to-all broadcast**: Each node sends the **same message** to all others.
* **All-to-all personalized**: Each node sends a **unique message** to every other node.

**Use Case**:

* Broadcast: Sharing a global state.
* Personalized: Matrix transpose (each element has a specific destination).

**4. Explain the hypercube algorithm for one-to-all broadcast.**

**Answer**:

1. **Steps**: log⁡plog*p* (one per dimension).
2. **Process**:
   * Source sends data along the highest dimension (e.g., dimension 2 in 3D).
   * Nodes receiving data in step k*k* forward it along dimension k−1*k*−1.
3. **Example**: 3D hypercube (Figure 4.2):
   * Step 1: Node 0 → Node 1 (dimension 2).
   * Step 2: Nodes 0,1 → Nodes 2,3 (dimension 1).
   * Step 3: Nodes 0-3 → Nodes 4-7 (dimension 0).

**5. What is the advantage of using a mesh for all-to-all personalized communication?**

**Answer**:

* **Phased Approach**:
  1. **Row-wise**: Exchange data within rows.
  2. **Column-wise**: Exchange data within columns.
* **Time**: O(p)*O*(*p*​) for p×p*p*​×*p*​ mesh vs. O(p)*O*(*p*) for ring.
* **Avoids Congestion**: Uses multiple paths simultaneously.

**6. How is all-reduce different from all-to-one reduction?**

**Answer**:

* **All-to-one reduction**: Results accumulated at **one node**.
* **All-reduce**: **All nodes** get the combined result.
* **Implementation**: All-reduce can use all-to-all broadcast pattern with arithmetic operations (e.g., sum).

**Example**: All nodes compute the global sum of their local values.

**7. Why does a circular shift on a mesh require two phases?**

**Answer**:

* **Phase 1 (Row Shift)**: Shift data within rows.
* **Phase 2 (Column Shift)**: Adjust for wraparound connections.
* **Reason**: Mesh lacks direct wraparound links; rows/columns are decoupled.

**Example**: 5-shift on 4×4 mesh (Figure 4.7):

* Node 0 → Node 1 (row) → Node 5 (column).

**8. Compare scatter and gather operations.**

**Answer**:

* **Scatter**: One-to-many; source distributes unique data.
* **Gather**: Many-to-one; root collects unique data.
* **Algorithm**: Gather reverses scatter’s steps (hypercube: log⁡plog*p* steps).

**9. What is the role of an associative operator in reduction?**

**Answer**:

* **Definition**: An operator ⊕⊕ where (a⊕b)⊕c=a⊕(b⊕c)(*a*⊕*b*)⊕*c*=*a*⊕(*b*⊕*c*).
* **Examples**: Addition, multiplication, max, min.
* **Importance**: Enables parallel tree-structured reduction.

**10. How does congestion occur in a linear array broadcast? How is it avoided?**

**Answer**:

* **Congestion**: If nodes send left/right simultaneously, links overlap (e.g., Node 1 → Node 2 and Node 2 → Node 1).
* **Solution**: Recursive doubling sends to **farthest nodes first** (e.g., Node 0 → Node 4 before Node 0 → Node 2).

**11. Explain the time complexity of all-to-all personalized communication on a hypercube.**

**Answer**:

* **Steps**: log⁡plog*p* (one per dimension).
* **Per Step**: Nodes exchange p/2*p*/2 messages.
* **Total Time**: m(p−1)*m*(*p*−1) for message size m*m*.

**Example**: 3D hypercube (Figure 4.6):

* Step 1: Exchange between subcubes (dimension 2).
* Step 2: Exchange within subcubes (dimension 1).
* Step 3: Final exchange (dimension 0).

**12. Why is the hypercube topology advantageous for communication operations?**

**Answer**:

* **Logarithmic Steps**: log⁡plog*p* for broadcast, reduction, etc.
* **Congestion-Free**: Each dimension provides disjoint paths.
* **Generalizability**: Algorithms extend to meshes/rings.

**13. What is the purpose of a circular shift operation? Provide an application.**

**Answer**:

* **Purpose**: Rotate data by q*q* positions.
* **Application**:
  + **Matrix Multiplication**: Shift rows/columns for Cannon’s algorithm.
  + **Pattern Matching**: Rotate strings to compare substrings.

**14. Describe the two-phase mesh algorithm for all-to-all broadcast.**

**Answer**:

1. **Phase 1 (Row Broadcast)**: Each node broadcasts within its row.
2. **Phase 2 (Column Broadcast)**: Each node broadcasts within its column.

* **Result**: All nodes receive data from all others.

**Time**: 2×O(p)2×*O*(*p*​) for p×p*p*​×*p*​ mesh.

**15. How does all-reduce leverage all-to-all broadcast’s communication pattern?**

**Answer**:

* **Same Steps**: log⁡plog*p* dimension-wise exchanges.
* **Difference**:
  + **All-to-all broadcast**: Concatenates messages.
  + **All-reduce**: Applies arithmetic operations (e.g., sum).
* **Efficiency**: Avoids separate reduction + broadcast steps.

**Example**: Hypercube all-reduce (Figure 4.3) sums values in log⁡plog*p* steps.

**Summary Table: Communication Operations**

| **Operation** | **Complexity (Hypercube)** | **Key Use Case** |
| --- | --- | --- |
| One-to-all broadcast | log⁡plog*p* | Distributing shared data |
| All-to-all personalized | m(p−1)*m*(*p*−1) | Matrix transpose |
| Scatter/gather | log⁡plog*p* | Distributing/collecting chunks |
| All-reduce | log⁡plog*p* | Global sum/max |
| Circular shift | O(p)*O*(*p*​) (mesh) | Matrix rotations |