CONSTRUCTING INDEPENDENT SPANNING TREES IN BUBBLE SORT NETWORKS

Presented By:

Sana Mir (22I-1160)

Ayesha Kiani (22I-1283)

Abdulrehman (22I-1182)

OUTLINE

1. **Problem Addressed**
2. **Background Concepts**
3. **Proposed Algorithm**
4. **Results & Analysis**
5. **Three Layer Architecture**

THE PROBLEM

- Constructing multiple independent spanning trees (ISTs) in bubble-sort networks
- Need for parallel algorithm where each vertex can determine its parent instantly
- Solving an open problem from previous research by Kao et al.

IMPORTANCE

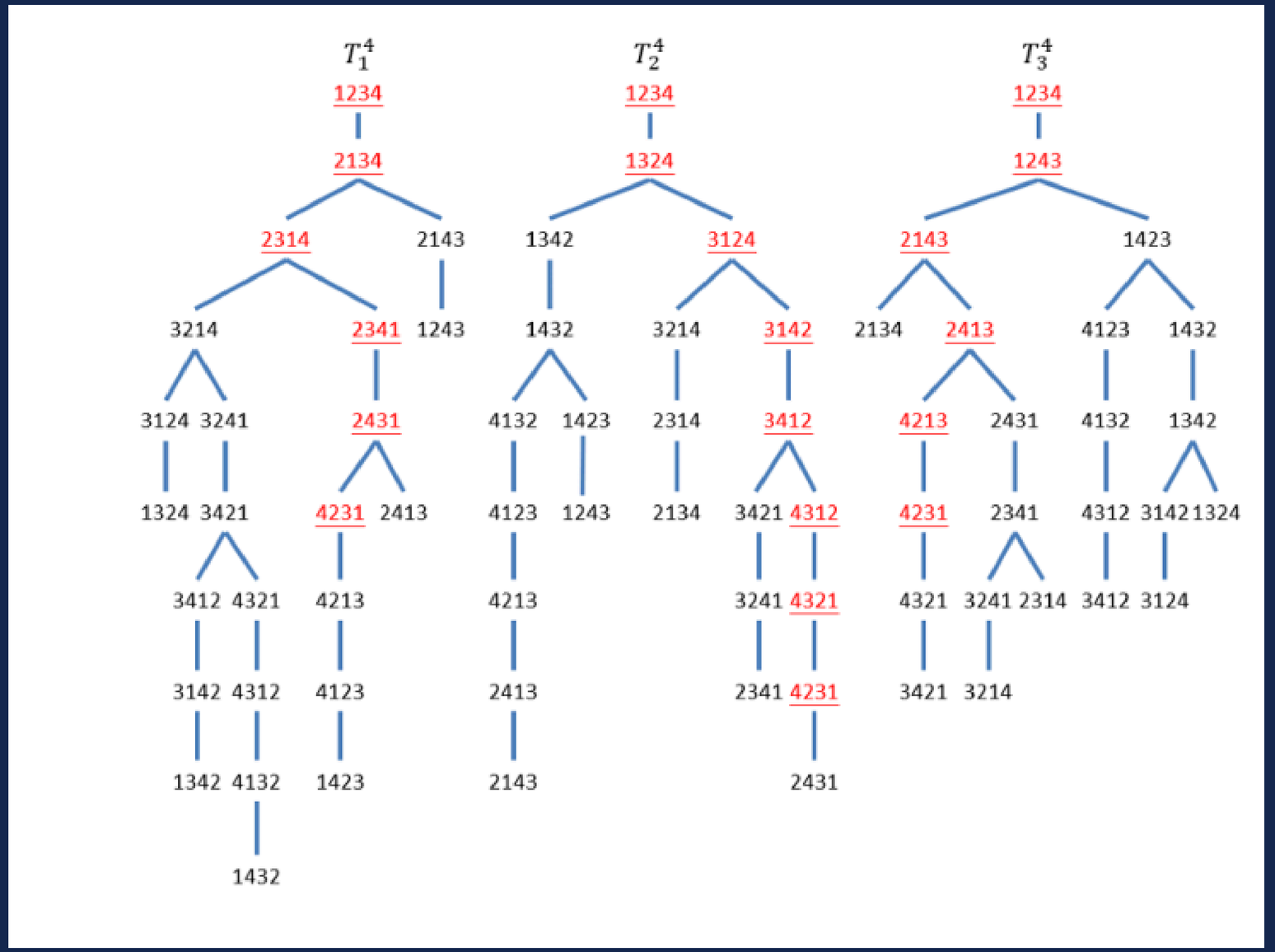
- ISTs provide fault-tolerant communication in networks
- Enable secure message distribution by splitting across independent paths
- Enhance reliability of interconnection networks
- Require no common edges or vertices between paths (except endpoints)

BUBBLE SORT NETWORKS

- Network based on permutations of numbers (like 1234, 4231)
- Connections exist between permutations that differ by one adjacent swap
- Highly fault-tolerant: Need to break n-1 connections to disconnect

PROPOSED ALGORITHM

- Non-recursive approach where each vertex independently finds its parent
- Rules based on vertex's last digits and which tree is being built
- Each vertex computes its parent in constant time
- Creates n-1 completely independent trees rooted at identity permutation

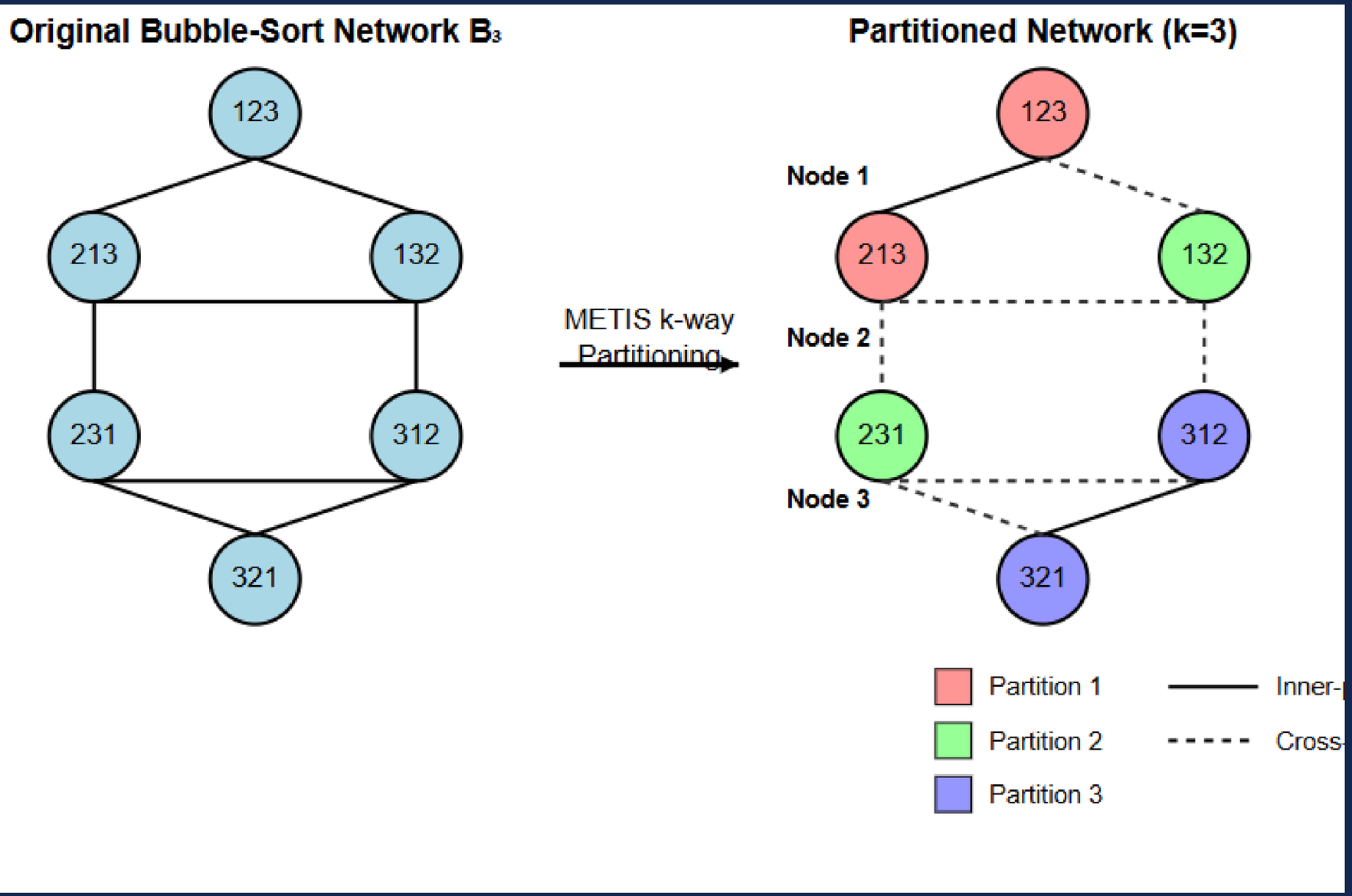

T_1^4
1234
2134
2314 2143
3214 2341 1243
3124 3241 2431
1324 3421 4231 2413
3412 4321 4213
3142 4312 4123
1342 4132 1423
1432
T_2^4
1234
1324
1342 3124
1432 3214 3142
4132 1423 2314 3412
4123 1243 2134 3421 4312
4213 3241 4321
2413 2341 4231
2143 2431
T_3^4
1234
1243
2143 1423
2134 2413 4123 1432
4213 2431 4132 1342
4231 2341 4312 3142 1324
4321 3241 2314 3412 3124
3421 3214

RESULT AND ANALYSIS

- Successfully constructs n-1 ISTs in bubble-sort networks
- Optimal total time complexity: $O(n \cdot n!)$
- Each vertex determines its parent in constant time
- Trees have height at most $D(B_n) + n-1$

METIS GRAPH PARTITIONING

- Bubble-sort networks grow factorially (n! vertices) - must be distributed
- METIS divides the network efficiently across computing nodes:
- Partitions vertices to balance computational load
- Minimizes edge cuts between partitions
- Preserves locality of related permutations when possible


Original Bubble-Sort Network B3
123
213
132
231
312
321
METIS k-way Partitioning
Partitioned Network (k=3)
Node 1
Node 2
Node 3
123
213
132
231
312
321
Partition 1
Partition 2
Partition 3
Inner-
Cross-

INTER-NODE MPI COMM

- Used for communication between computing nodes
- Each node processes its assigned partition of the bubble-sort network
- MPI_Send/MPI_Recv for boundary vertex information exchange
- Collective operations (MPI_Gather) to compile final tree structures

OPENMP FOR INTRA-NODE PARALLELISM

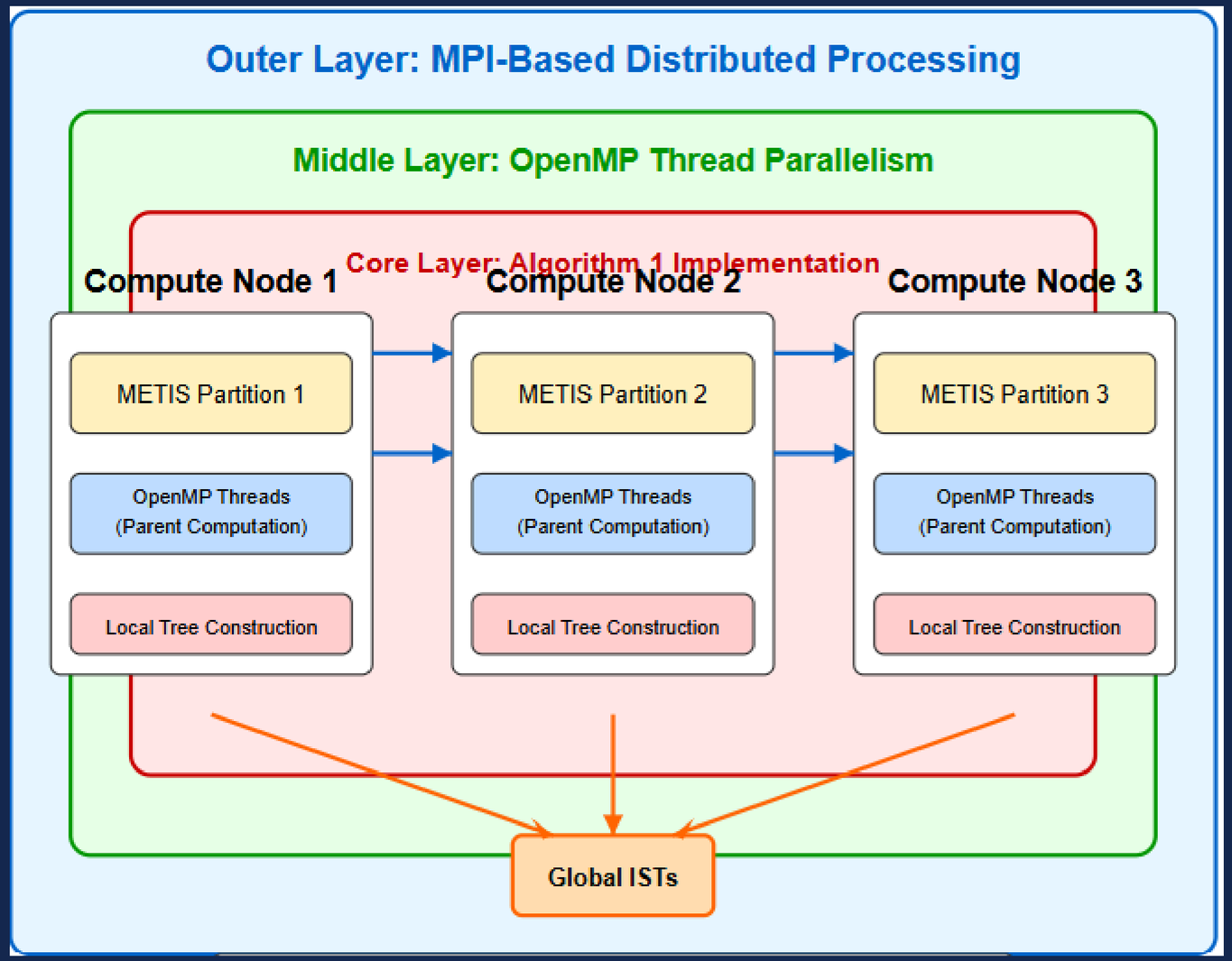
- Used for parallelism within each computing node
- Parallelize vertex processing using #pragma omp parallel for
- Each thread computes parents for a subset of vertices
- No dependencies between vertices makes this highly efficient
- Shared memory model simplifies data access

IMPLEMENTATION APPROACH

Pragma omp parallel for to distribute vertices among threads.
Thread-local storage for intermediate calculations.
No synchronization needed during primary computation.

```
#pragma omp parallel for
for(int i = 0; i < local_vertices; i++) {
    for(int t = 0; t < n-1; t++) {
        parent[i][t] = findParent(vertices[i], t, n);
    }
}
```

THREE LAYER ARCHITECTURE FOR IST CONSTRUCTION



CONCLUSION

- First fully parallel algorithm for this problem
- Solves an open problem in distributed systems
- Applications in fault-tolerant routing and secure communication
- Practical implementation through MPI+OpenMP with METIS partitioning

THANK YOU