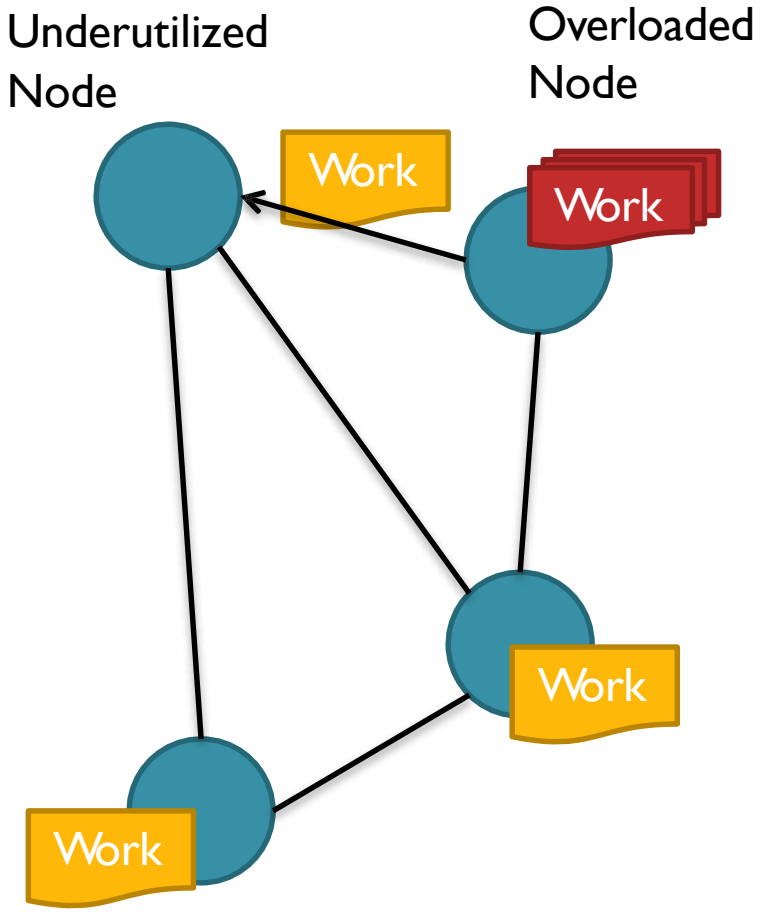
MP4: Dynamic Load Balancer

Problem Overview

- Underutilized Nodes
- Overloaded Nodes
- Heterogeneous Nodes
  - CPU
  - Memory
  - Battery
- Heterogeneous Workloads
- Transient Variations



Solution: Move Data to maintain even Utilization across Nodes

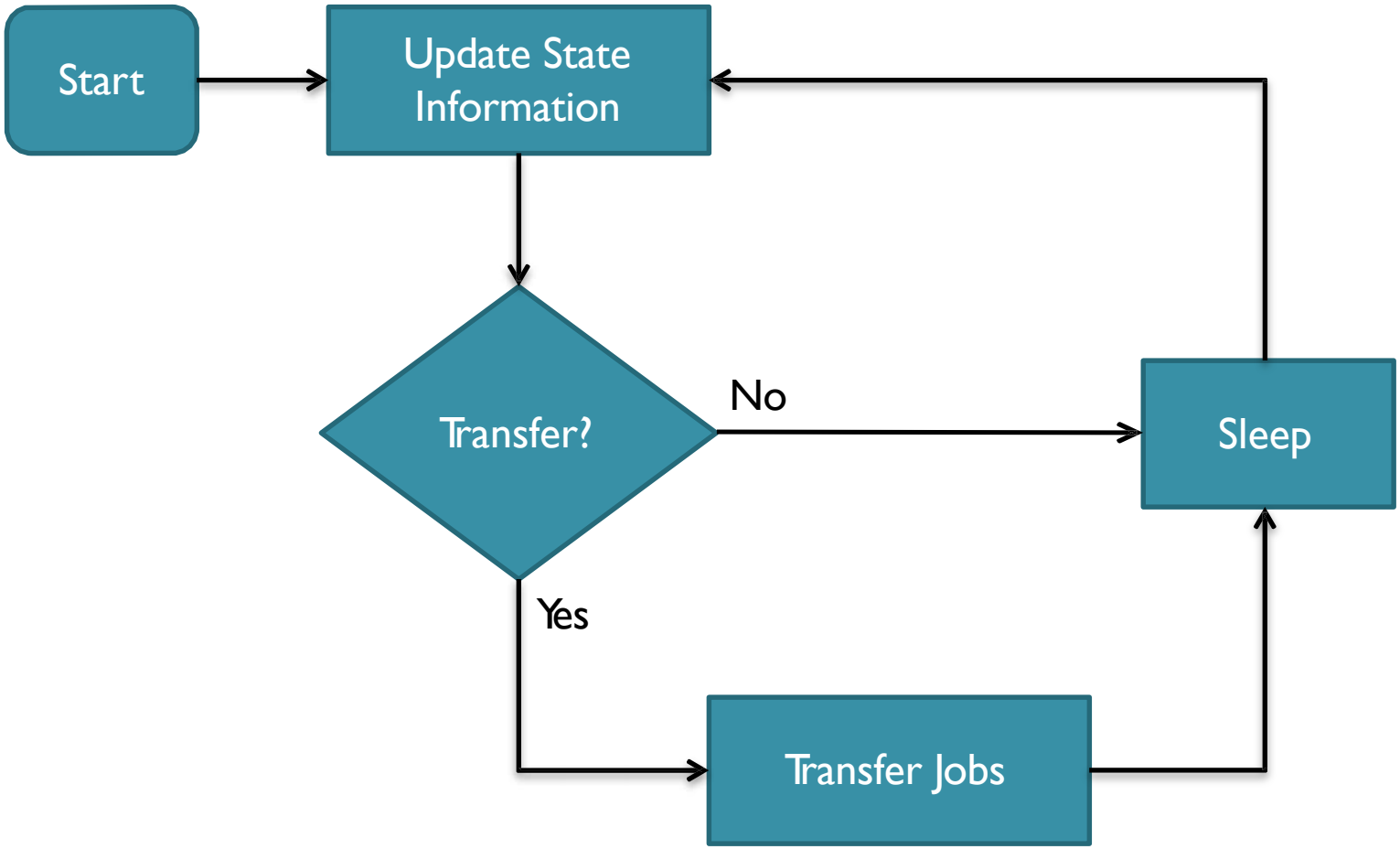
Algorithms Classification

MP4 Load Balancer

Parameters	Round Robin	Random	Local Queue	Central Queue	Central Manager	Threshold
Overload Rejection	No	No	Yes	Yes	No	No
Fault Tolerant	No	No	Yes	Yes	Yes	No
Forecasting Accuracy	More	More	Less	Less	More	More
Stability	Large	Large	Small	Small	Large	Large
Centralized/Decentralized	D	D	D	C	C	D
Dynamic/Static	S	S	Dy	Dy	S	S
Cooperative	No	No	Yes	Yes	Yes	Yes
Process Migration	No	No	Yes	No	No	No
Resource Utilization	Less	Less	More	Less	Less	Less

Dynamic Load Balancing

- Use System State Information at runtime



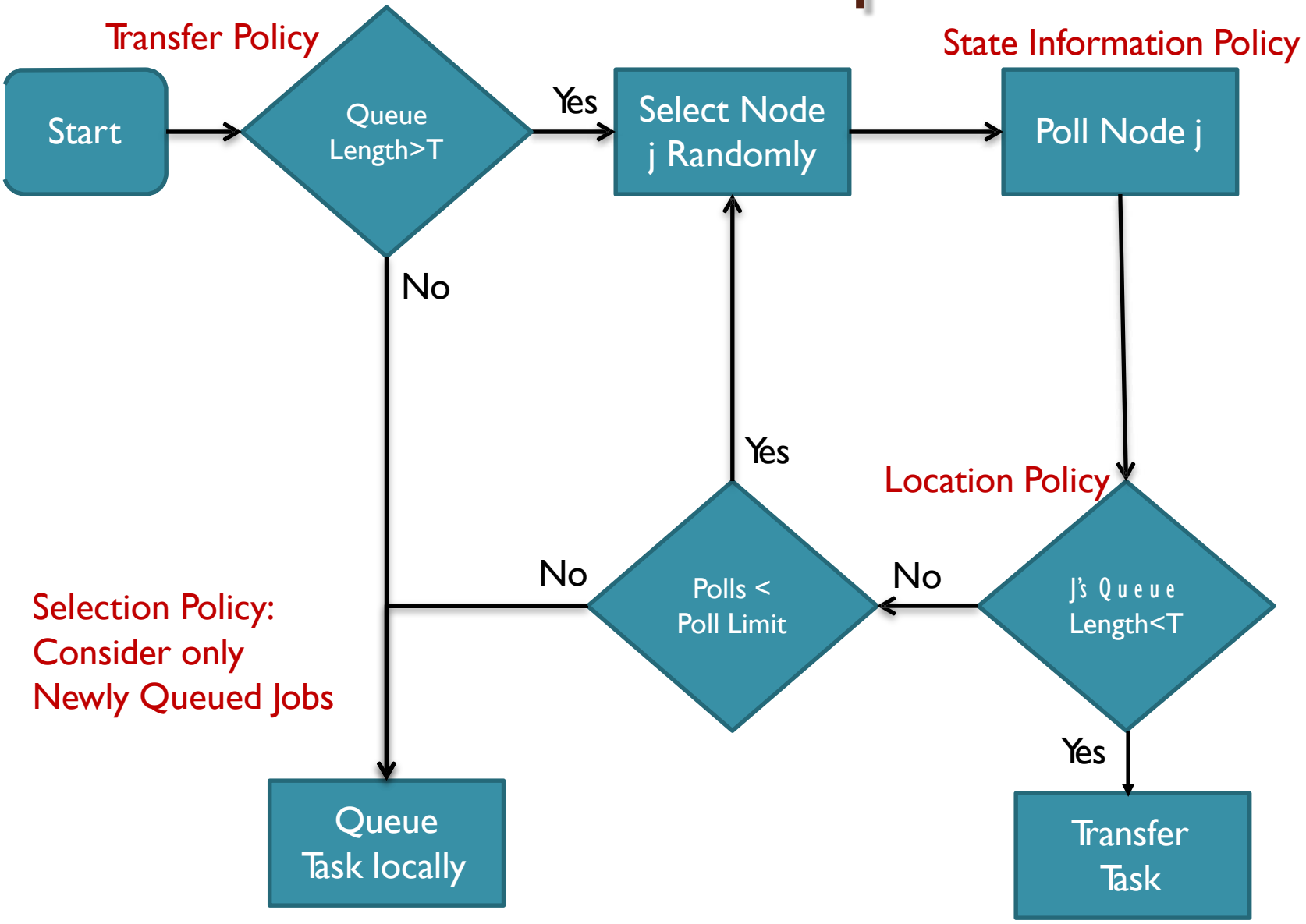
Policies

- Transfer Policy: When to initiate a transfer
  - Load Index: Metric used to decide over a transfer policy (e.g. Queue Length)
- Selection Policy: Which jobs will be transferred
- Location Policy: Partner node to receive or send jobs
  - Trivial for MP4!
- State Information Policy: How often to update state information
  - Time: Every 30 seconds
  - Events: When CPU usage is > 80%
  - Demand Driven: When needed by one policy

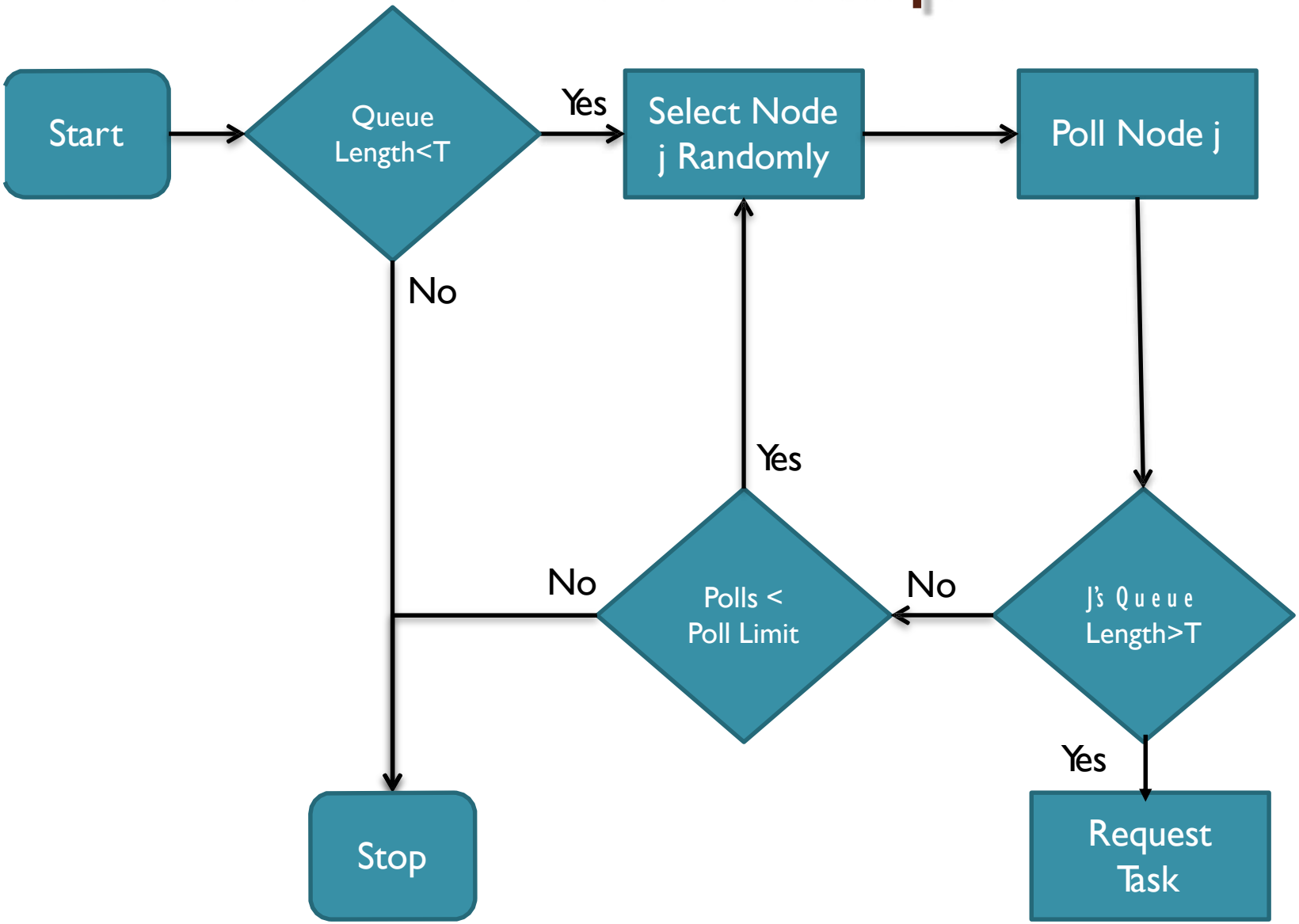
Transfer Direction

- Sender Initiated
  - Overloaded Nodes
- Receiver Inititated
  - Underloaded Nodes
- Symmetrically Initiated
  - Both but higher instability

Sender Initiated Example



Receiver Initiated Example



MP4 Overview

- Distributed Architecture of two nodes
  - Two Android Phones or Two Virtual Machines
    - Each VM in a separate Server



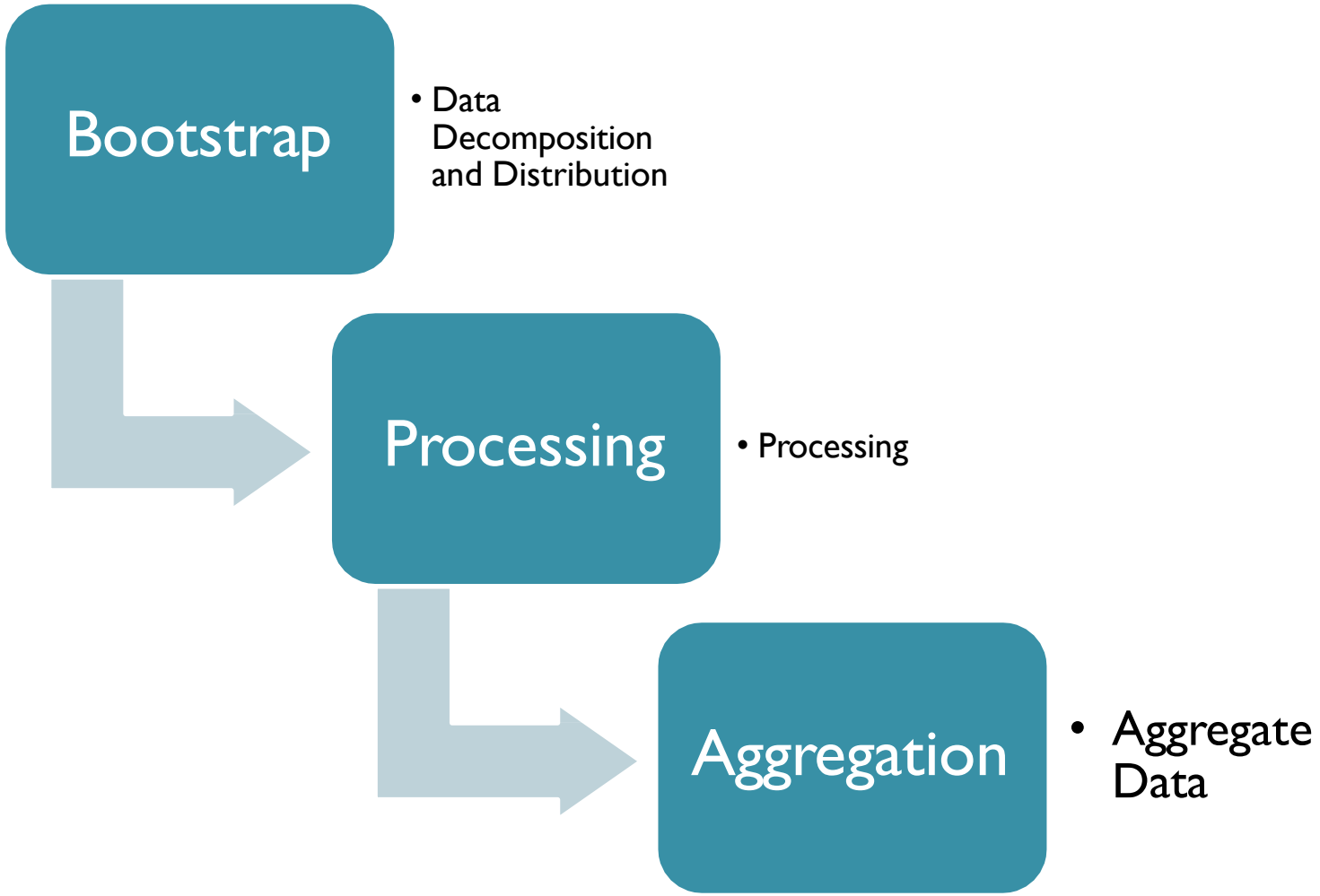
MP4 Overview

- Distributed Architecture of two nodes
  - Two Android Phones or Two Virtual Machines
- Implement a Dynamic Load Balancer
- Decide and implement each policy:
  - Transfer Policy
  - Selection Policy
  - State Information Policy
  - Location Policy Trivial!

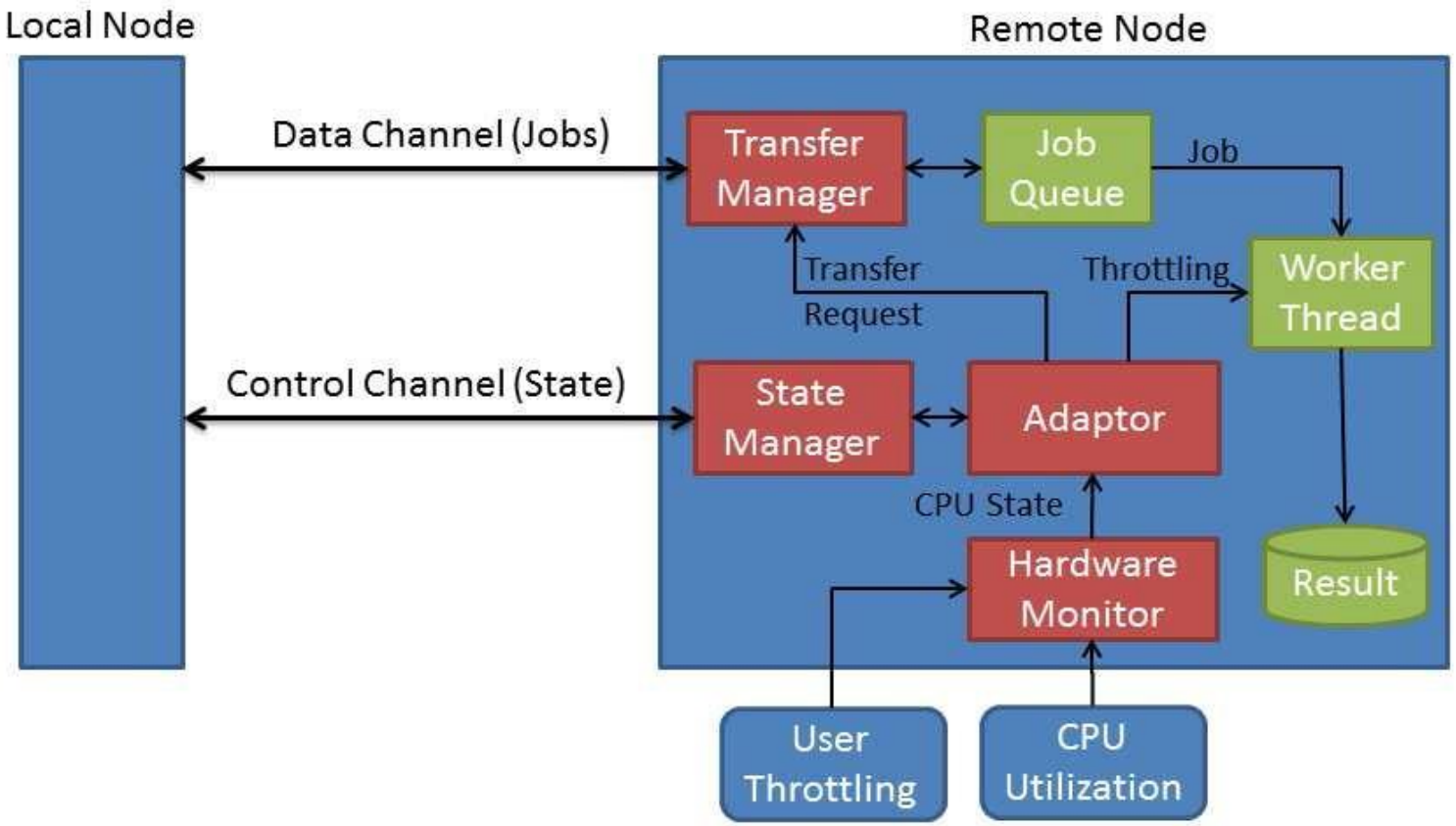
Workload Selection and Data Decomposition

- A simple vector addition
  - Purpose of this MP is to implement the loader balancer, not to come up with complicated workload
  - Vector addition on a 1024*1024*4 double vector
- Fully Parallelizable Algorithm
  - Jobs: Chunks of equal size
  - Initially there should be 512+ jobs

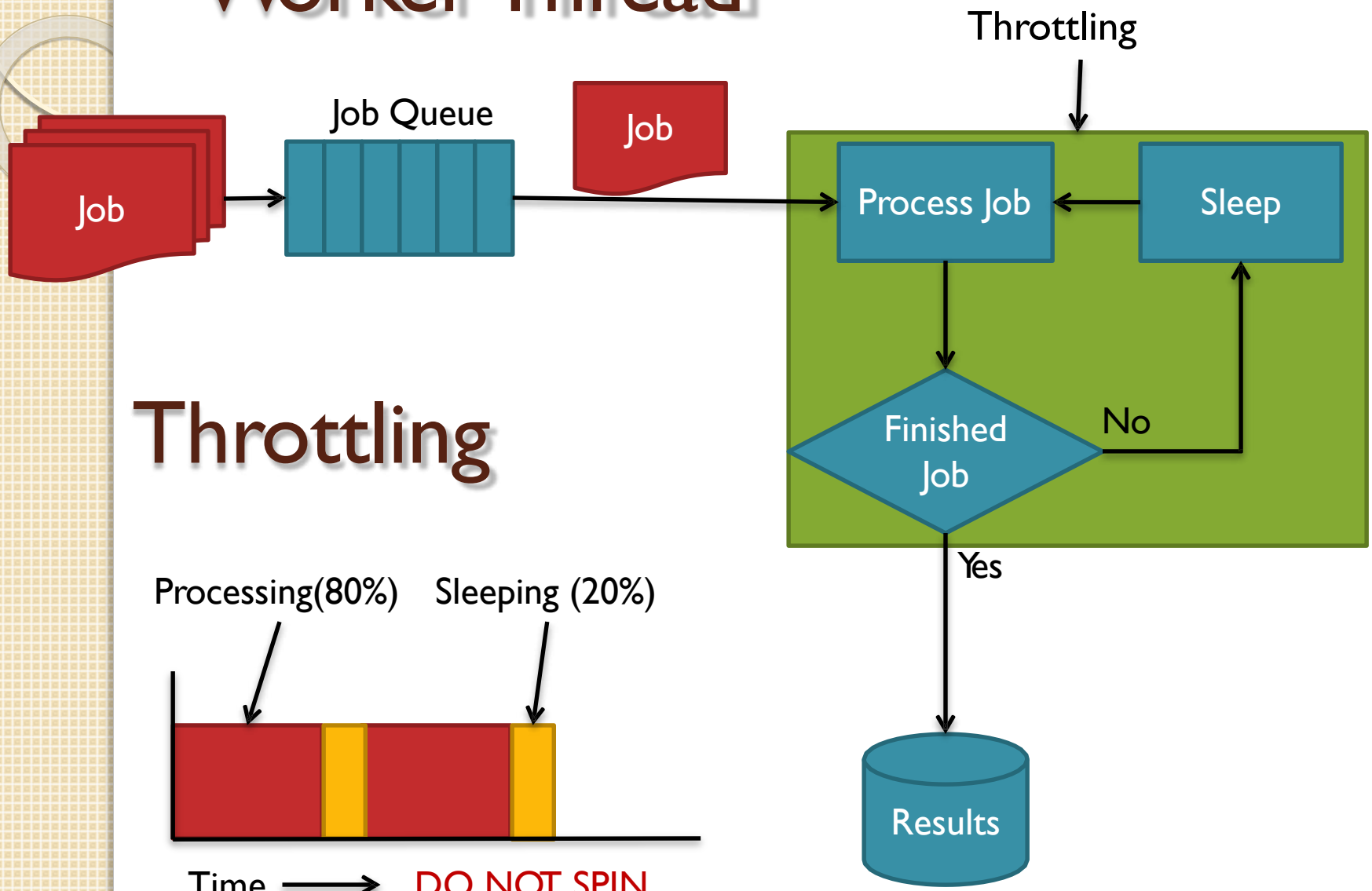
System Lifecycle



MP4 Minimal Architecture



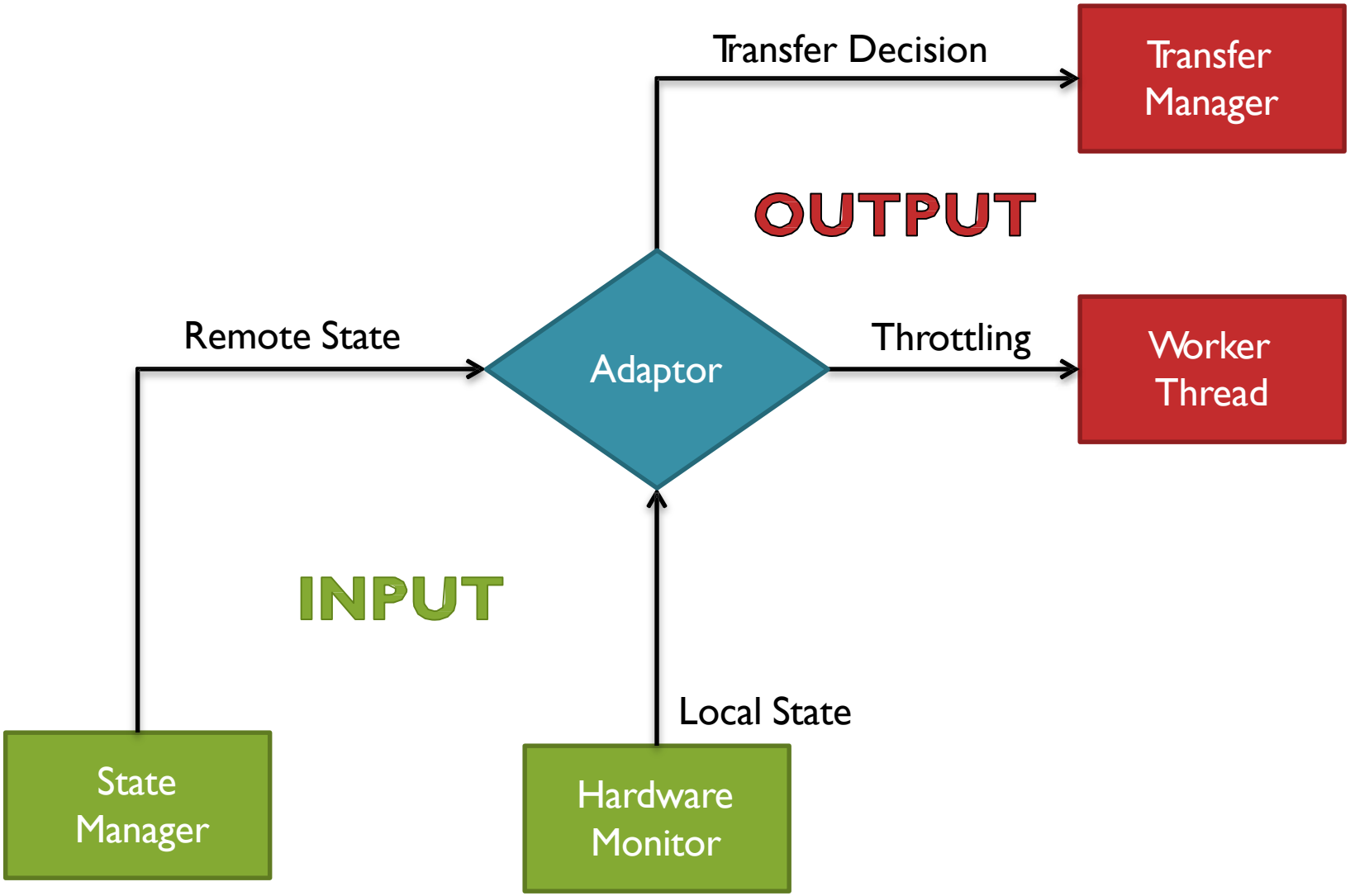
Worker Thread



Throttling

DO NOT SPIN
INSTEAD OF SLEEPING!

Load Balancer



Transfer Manager

- Transfer Jobs between the two phones
  - Use any protocol you decide
    - TCP/IP, UDP, HTTP, etc
- Tradeoffs:
  - Bandwidth
  - Stability
  - Performance

State Manager

- Transfer Local State to Remote phone
- Transfer Remote State to Local Phone
- Information State Policy
- Tradeoffs:
  - Stability
  - Bandwidth
  - Performance

Hardware Monitor

- Query system to monitor system state
  - CPU Usage
  - User Defined Throttling
  - Other Optional:
    - Battery State
    - Bandwidth
- Information State Policy
  - Less frequently vs More frequently
  - Event Based?
    - Android Event: Broadcast Listener

Implementation Guidelines

- Basic Architecture Required
  - Build on top of it
- Policy Design and Analysis
  - Justify, Measure and Validate
- Data Structures
  - Performance / Computational Complexity
- System Principles
  - Timers, Threads, Synchronization, Events
- Graphical User Interface
  - Runtime Parameters and Statistics

Improvements

- **Not required in this MP**
- Bandwidth (Transfer and State Managers)
- Information Policy Analysis
- Concurrent Worker Threads
- Graphical User Interface
- Compression