EE382N Garg: Distributed Systems

A Distributed Algorithm for Minimizing Travel Time in Traffic Grids

Chinwei “Vic” Hu

Saddam Quirem

Advisor: Dr. Vijay Garg

Outline

Introduction

Model of the Problem

Algorithms

Demo

Results & Discussion

Conclusion

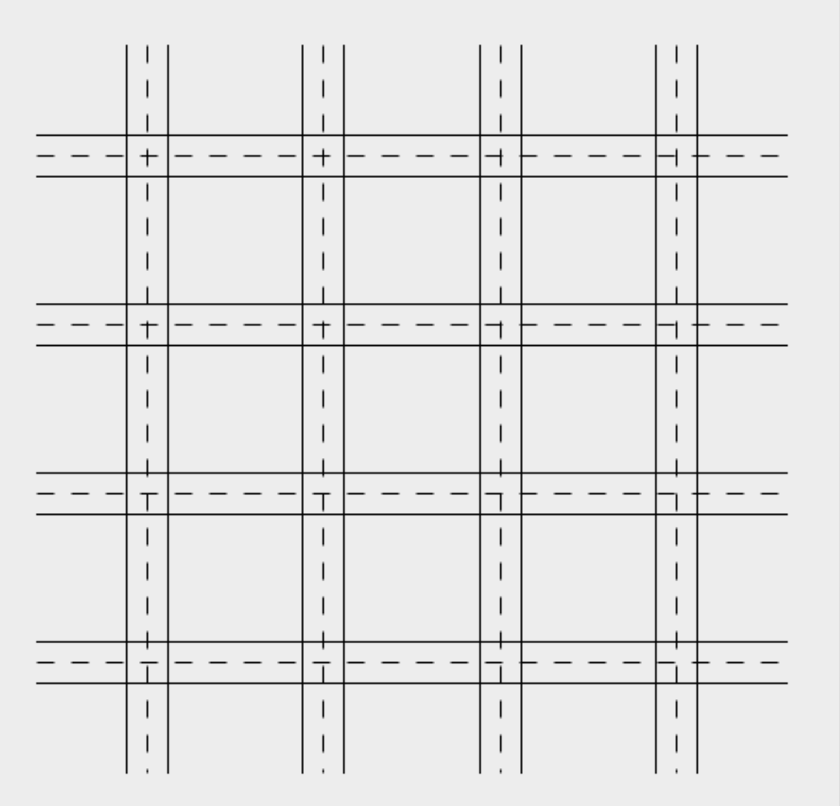
STOP
WRONG WAY
DO NOT ENTER

Introduction

- Motivation
- Problem Definition
- Solution Intuition

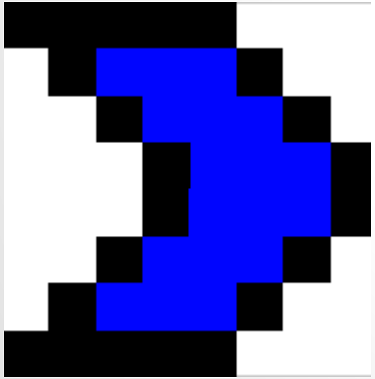
Model

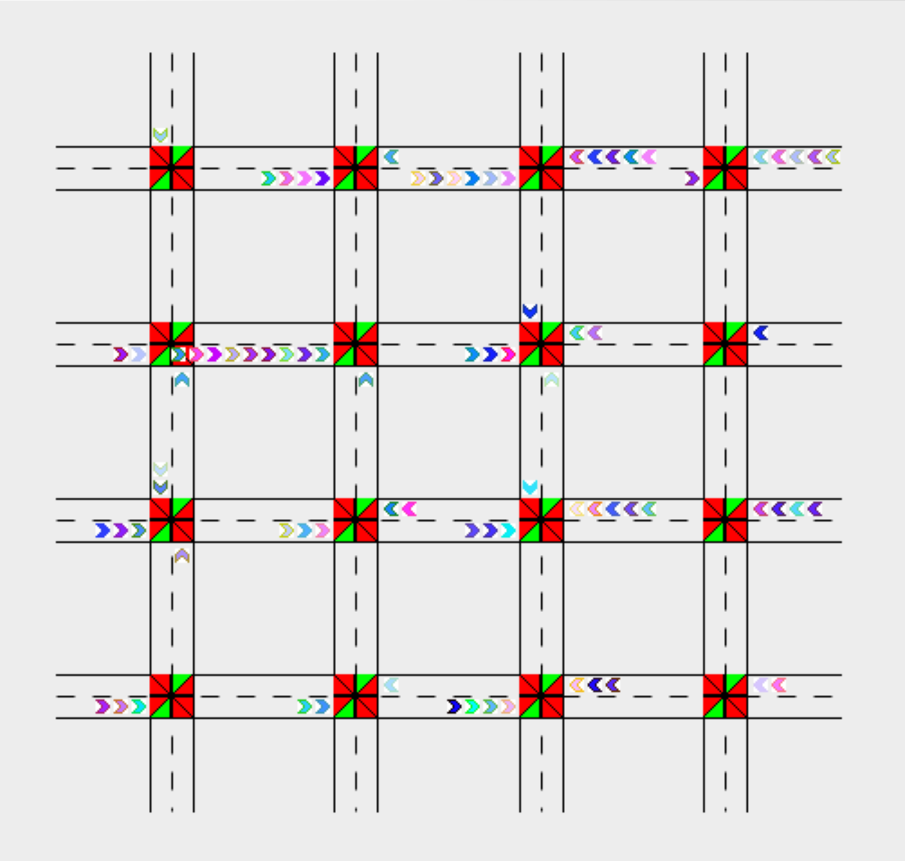
- GridWorld: W x H



Model

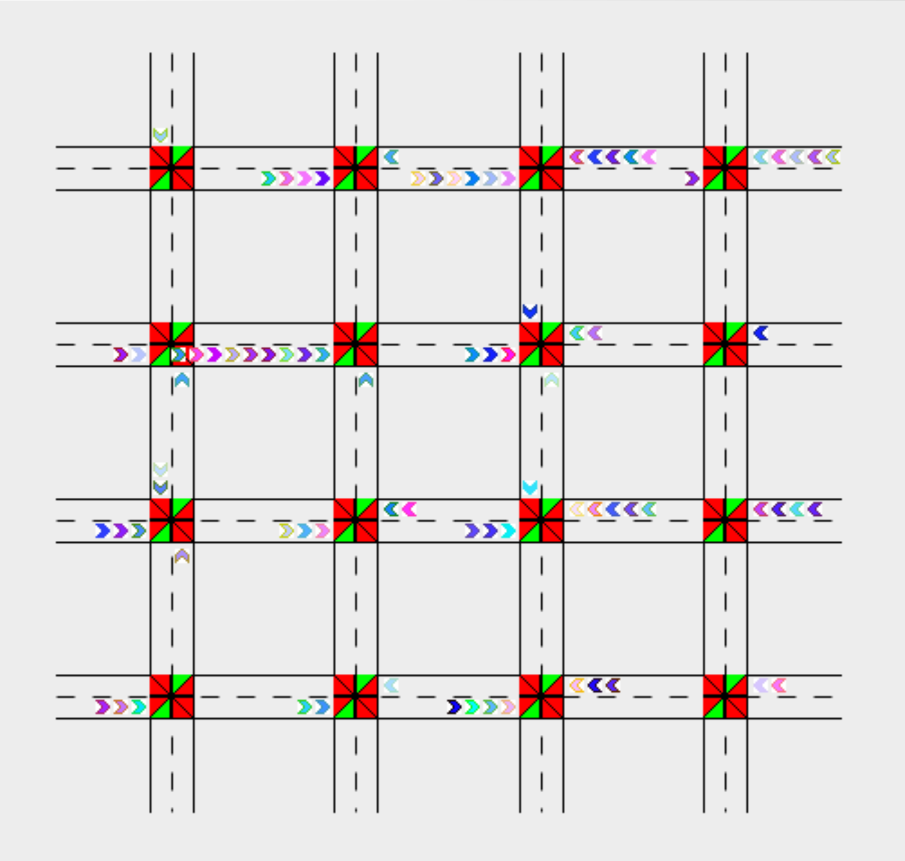
- GridWorld: W x H
- N: vehicle count





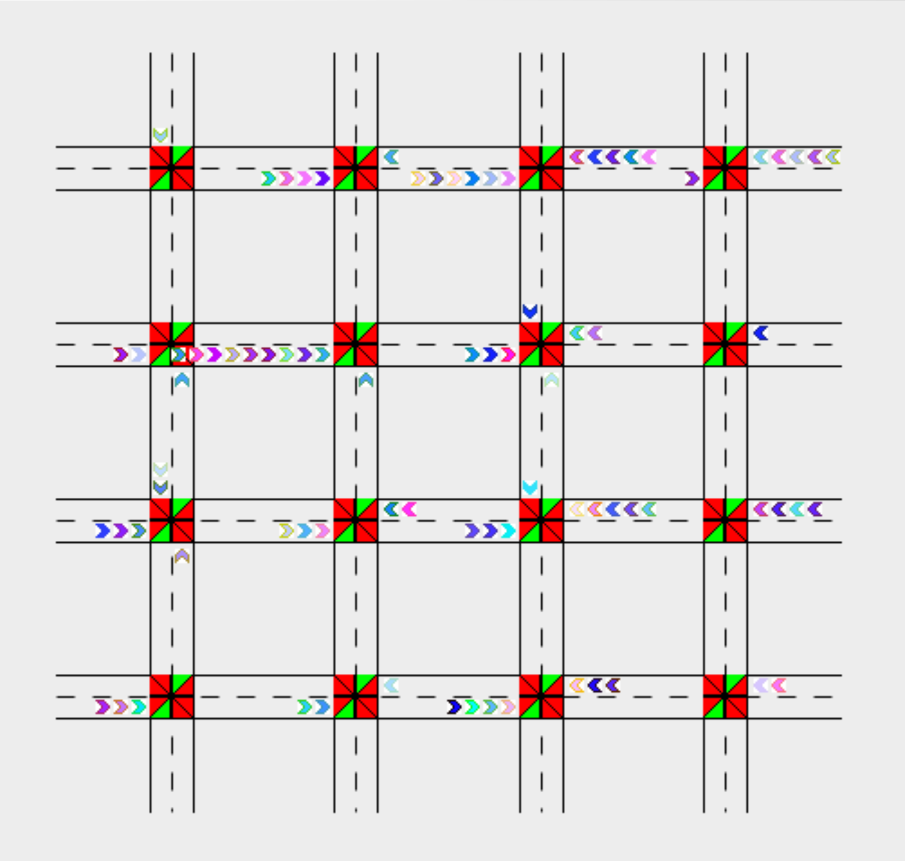
Model

- GridWorld: W x H
- N: vehicle count
- Double lane, 4 ways



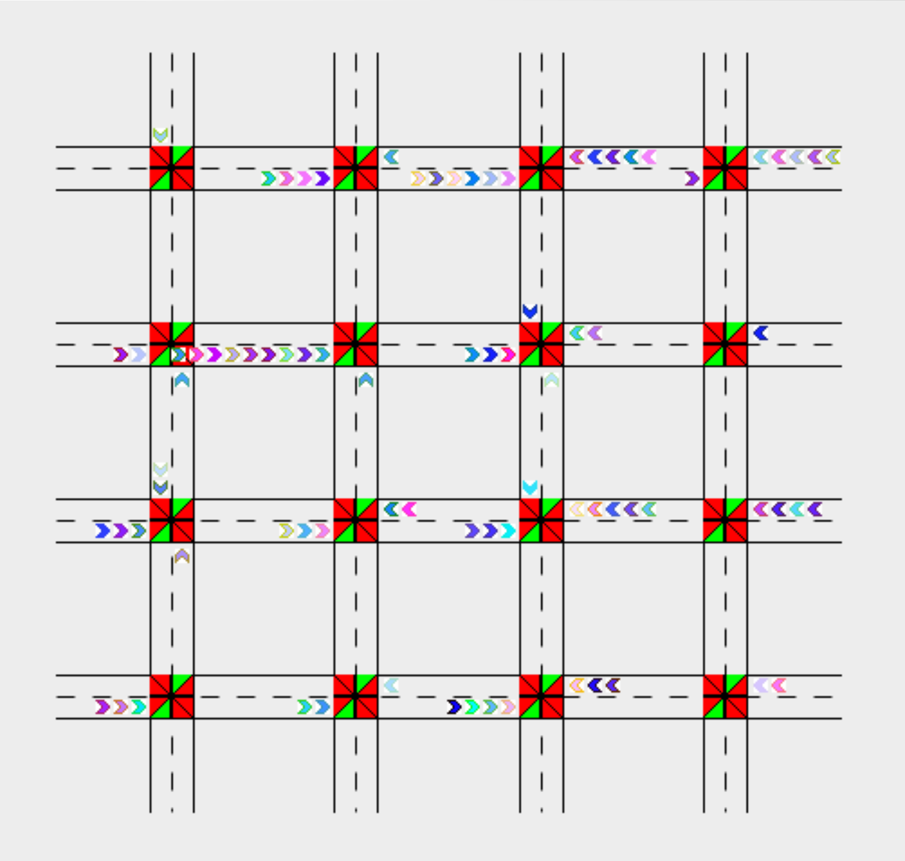
Model

- GridWorld: W x H
- N: vehicle count
- Double lane, 4 ways
- Client-Server architecture

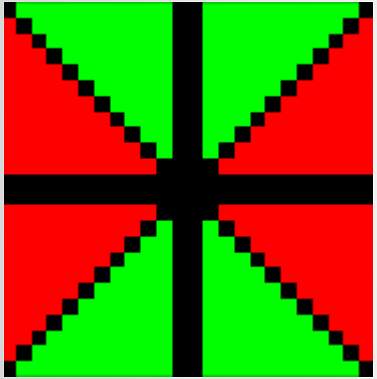


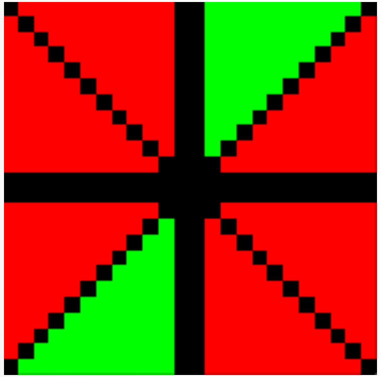
Model

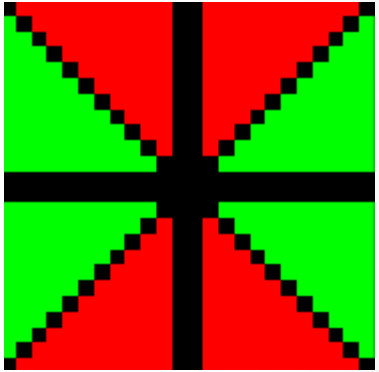
- GridWorld: W x H
- N: vehicle count
- Double lane, 4 ways
- Client-Server architecture
- 4 intersection states

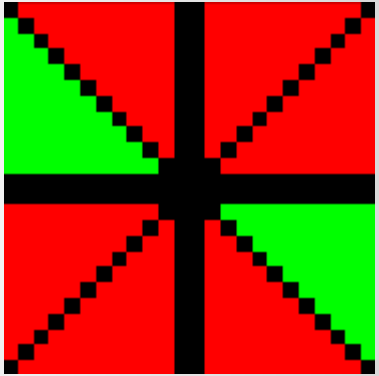


Intersection States

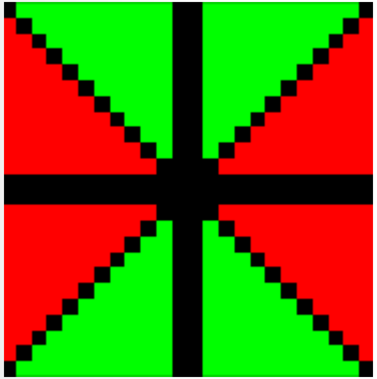


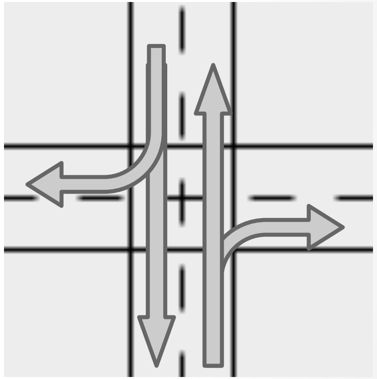




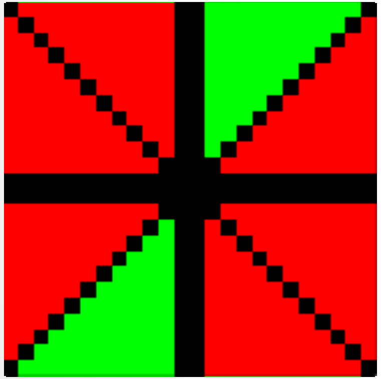


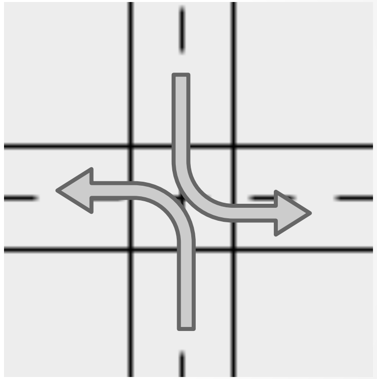
VERTICAL_STRAIGHT



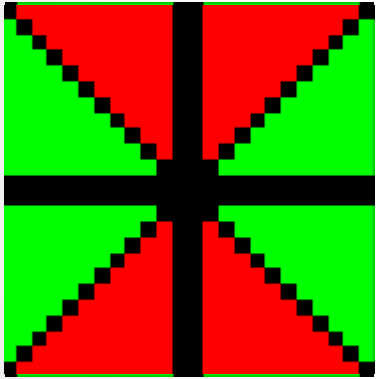


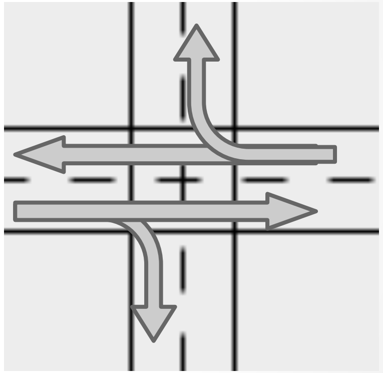
VERTICAL_LEFT



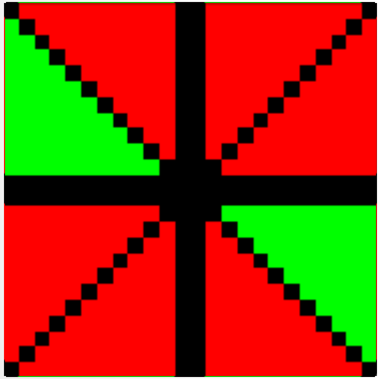


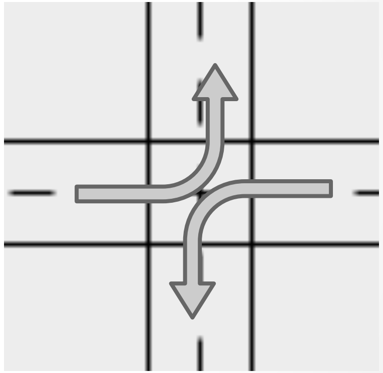
HORIZONTAL_STRAIGHT



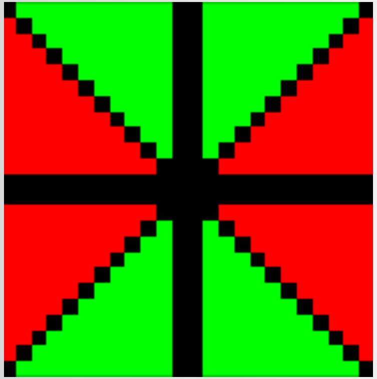


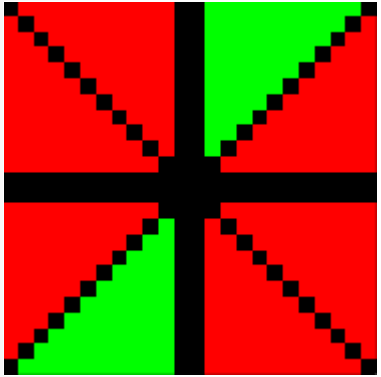
HORIZONTAL_LEFT

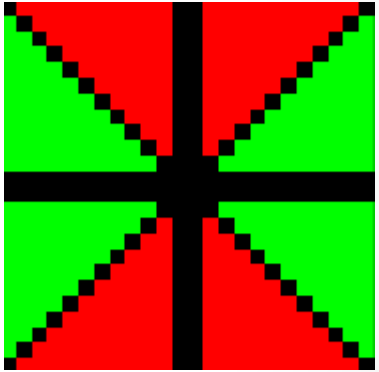


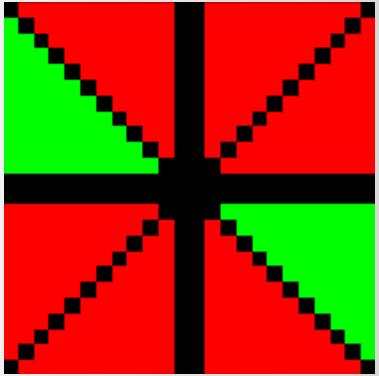


Intersection States



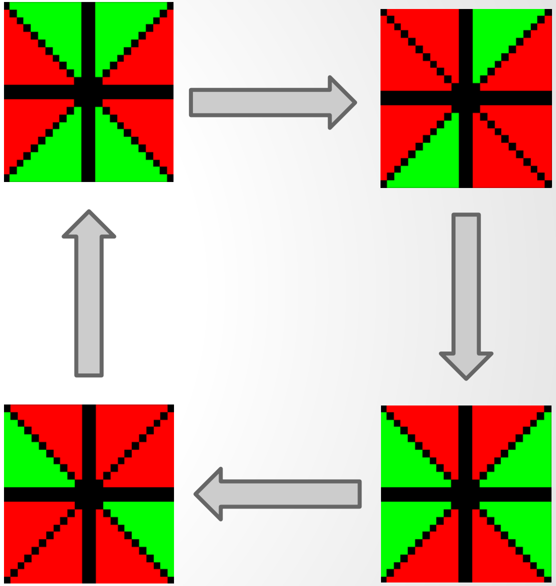






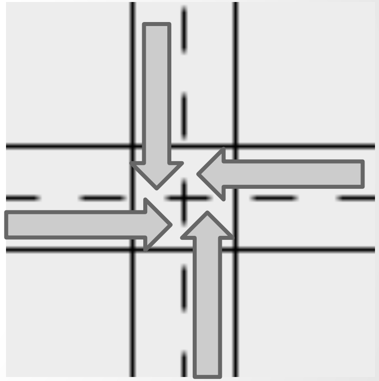
Dummy Algorithm

- A static state machine



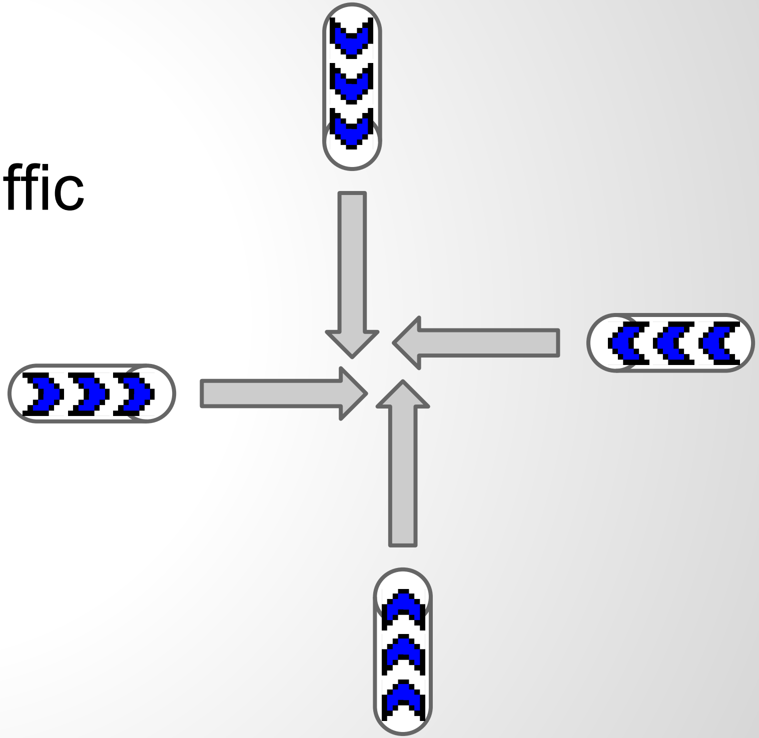
Dummy Algorithm

- A static state machine
- Keeps one queue for each traffic direction



Dummy Algorithm

- A static state machine
- Keeps one queue for each traffic direction



Dummy Algorithm

- A static state machine
- Keeps one queue for each traffic direction
- The intersection grants request based on the current state
    - **VERTICAL**: NORTH & SOUTH
    - **HORIZONTAL**: WEST & EAST

Dummy Algorithm

- A static state machine
- Keeps one queue for each traffic direction
- The intersection grants request based on the current state
    - **VERTICAL**: NORTH & SOUTH
    - **HORIZONTAL**: WEST & EAST
    - **LEFT**: Check both fronts are turning *left*
    - **STRAIGHT**: Check both fronts are going *straight* or *left*

Dummy Algorithm

Pros:

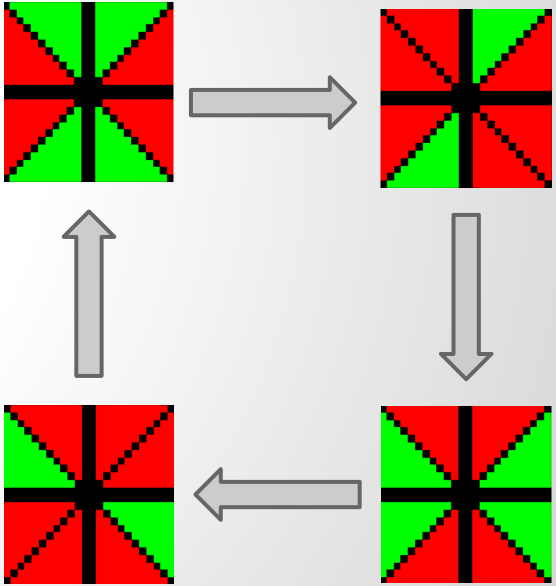
- Easy to implement
- Low computation overhead

Cons:

- Dumb
- Wasted cycles
- Wasted resources

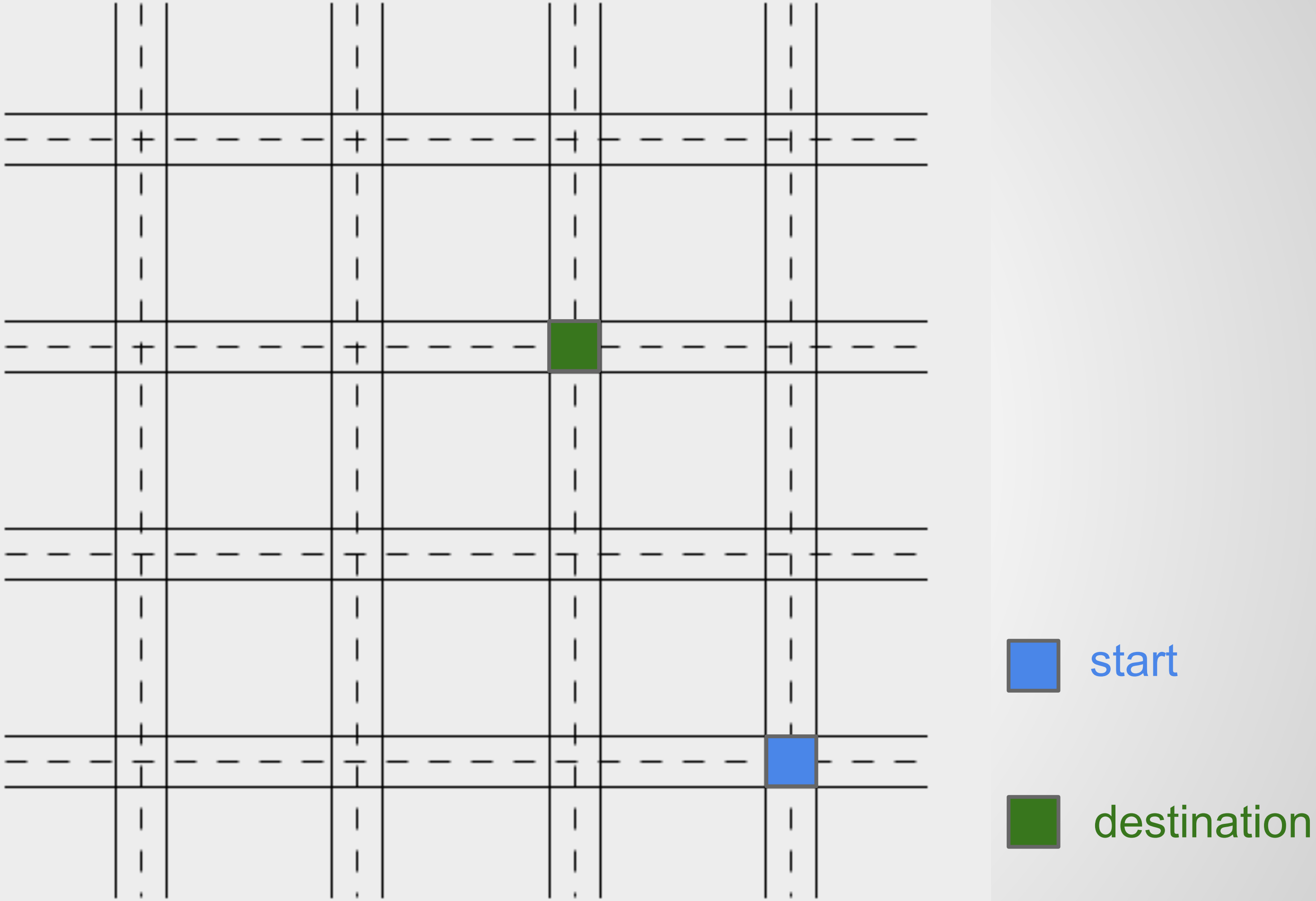
LookAhead Algorithm

- Same state machines as in Dummy



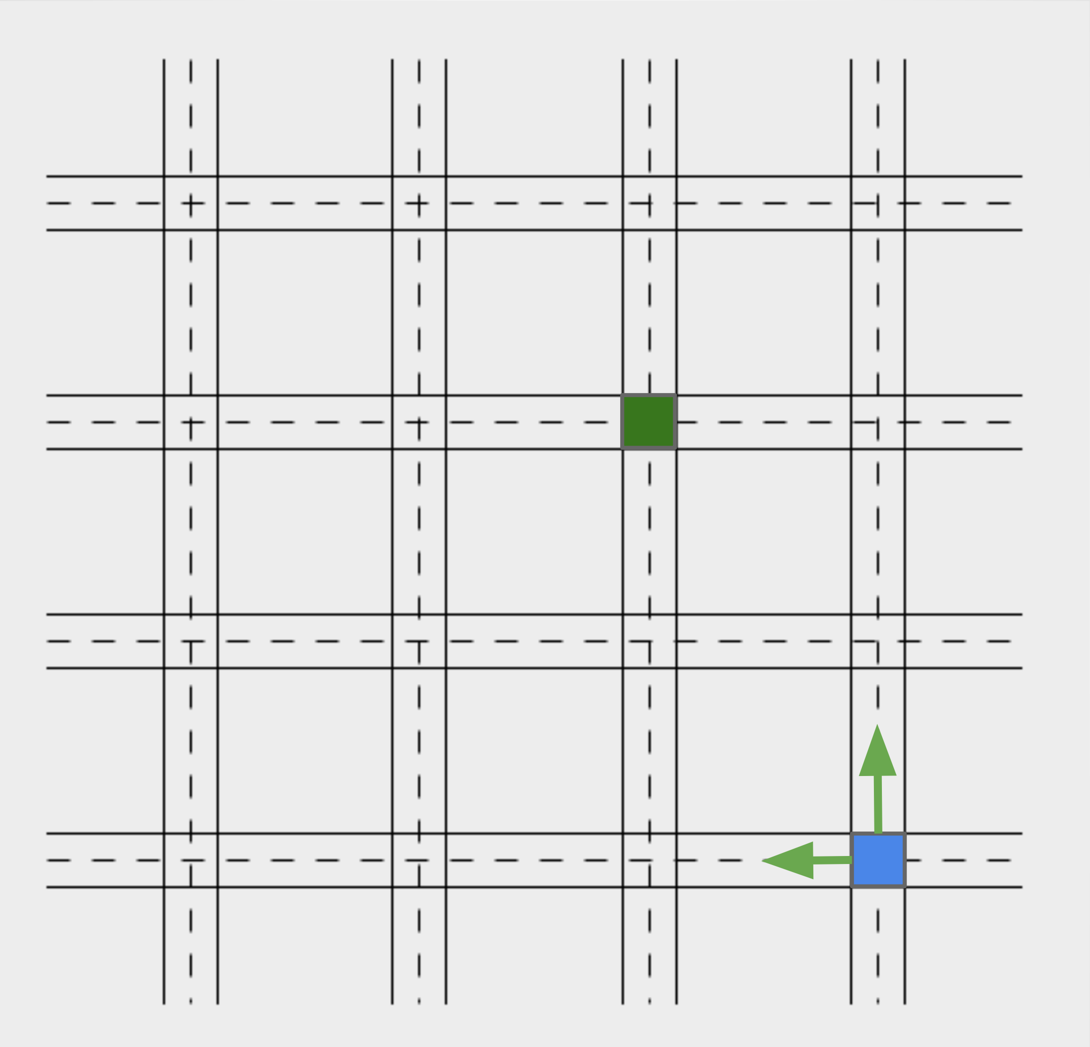
LookAhead Algorithm

- Same state machines as in Dummy
- Proactively suggests the optimal action for the vehicle



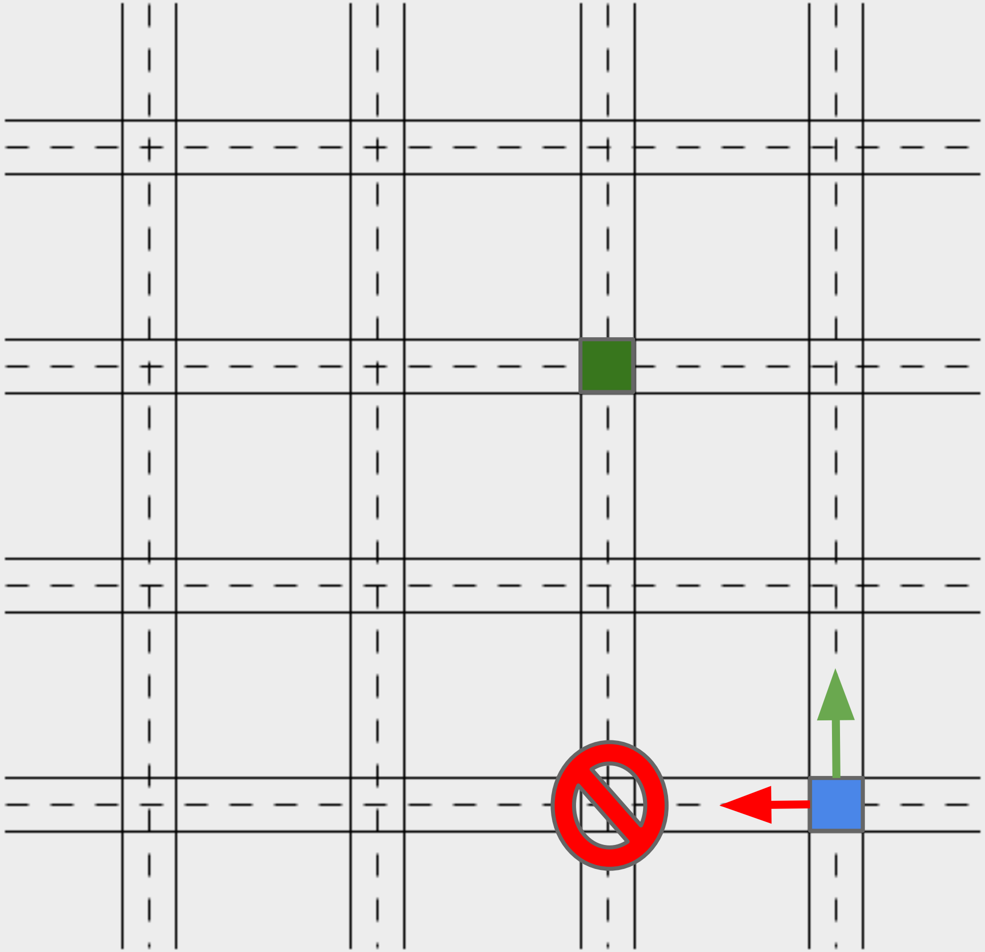
🟦 start

🟩 destination



🟦 start

🟩 destination



🟦 start

🟩 destination

LookAhead Algorithm

- Same state machines as in Dummy
- Proactively suggests the optimal action for the vehicle
- The optimal action is determined by the queue size of the future intersection

LookAhead Algorithm

Pros:

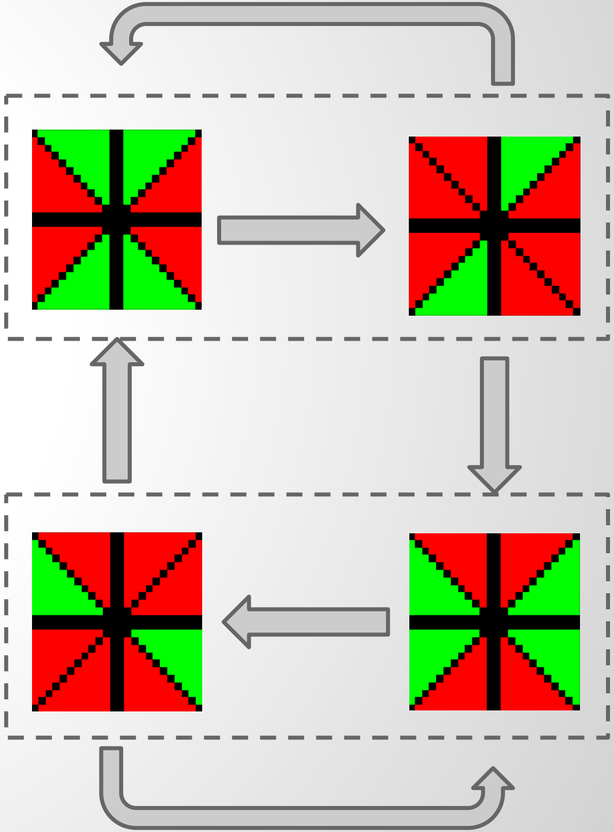
- Effectively avoid immediate gridlocks
- Reasonable computational overhead

Cons:

- Requires individual instructions
- Same wastes for synchronous design

WeightedQ-Switcher Algorithm

- Dynamic state machine



WeightedQ-Switcher Algorithm

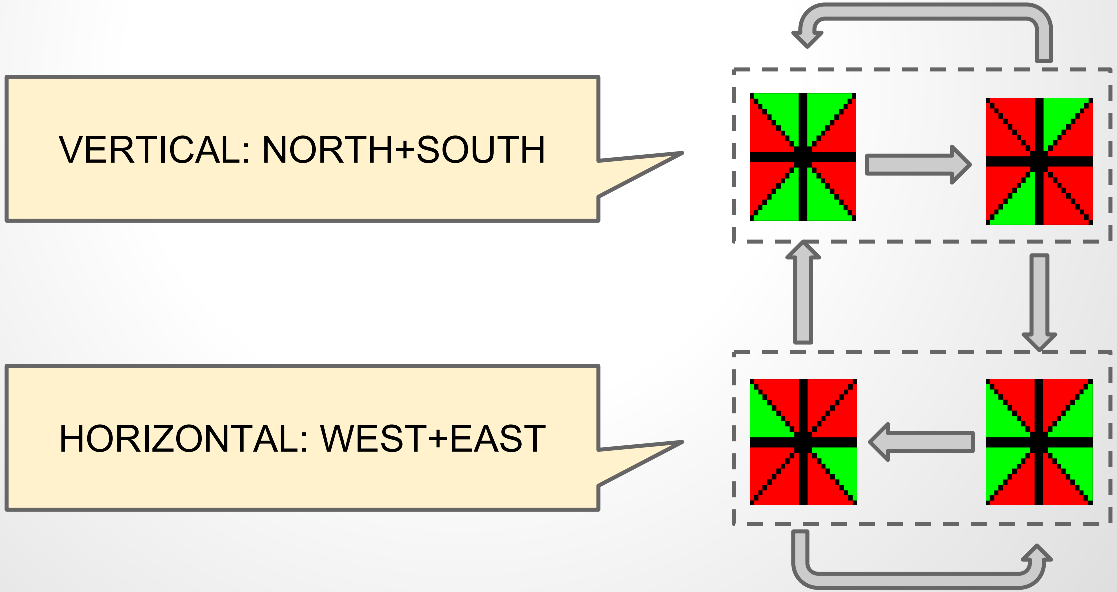
- Dynamic state machine
- State transitions based on queue weights

WeightedQ-Switcher Algorithm

- Dynamic state machine
- State transitions based on queue weights
    - Each request in the queue keeps a counter
    - Queue weights are calculated based on the sum of these counters

WeightedQ-Switcher Algorithm

- Dynamic state machine
- State transitions based on queue weights

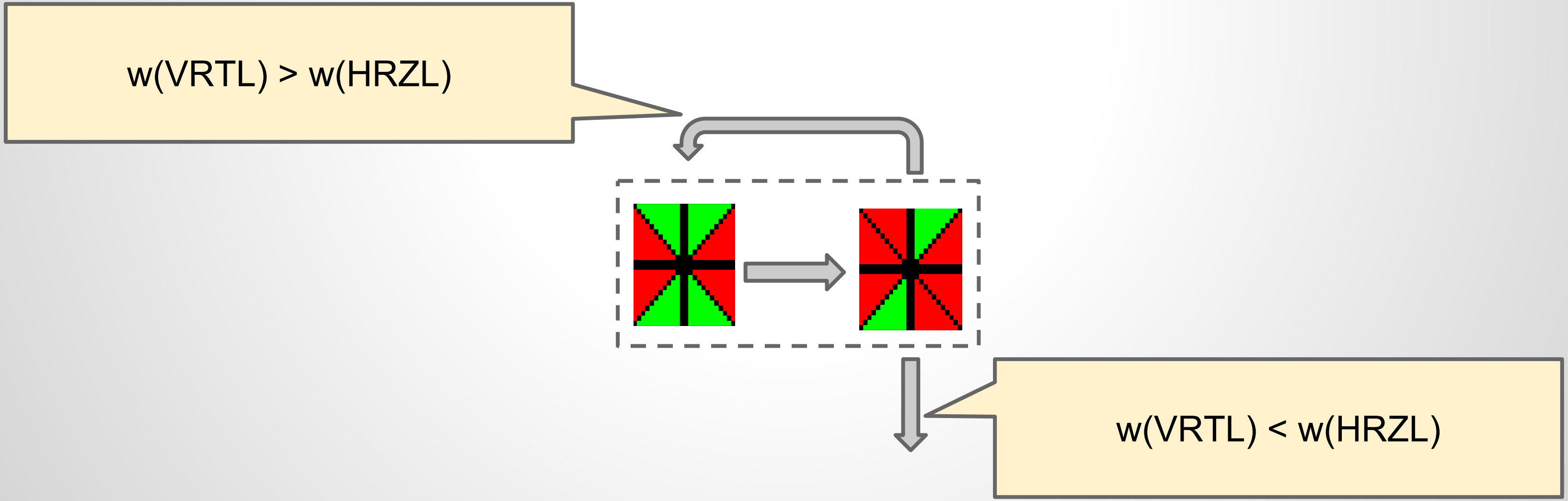


VERTICAL: NORTH+SOUTH

HORIZONTAL: WEST+EAST

WeightedQ-Switcher Algorithm

- Dynamic state machine
- State transitions based on queue weights



w(VRTL) > w(HRZL)

w(VRTL) < w(HRZL)

WeightedQ-Switcher Algorithm

Pros:

- Responsive to real-time traffic
- Easily integratable to existing infrastructure

Cons:

- computational overhead

LA-WQS Algorithm

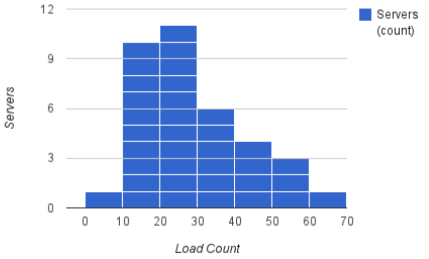
- Dynamic state transition + Optimal action routing

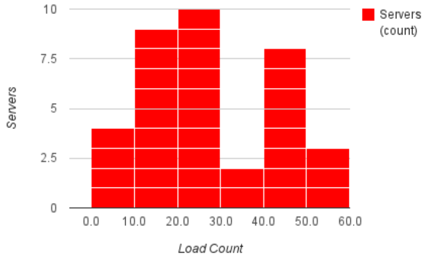
Demo

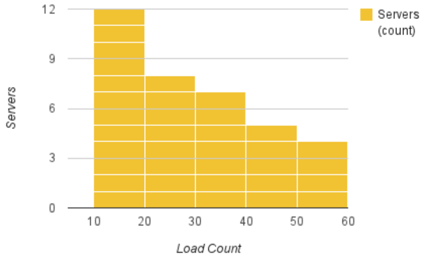
Experiments

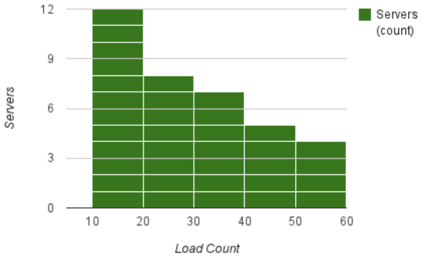
Simulation were ran with 256 and 512 vehicles and grid sizes ranging from 4x4 to 8x8, using four different algorithms, for a total of 72 different simulations.

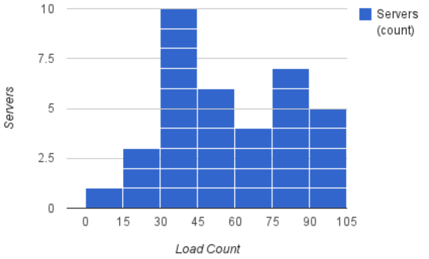
Results: Server Workload

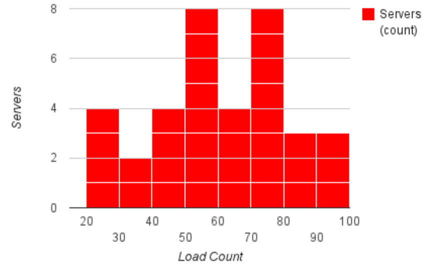

Distribution of Server Load (DUMMY-256)

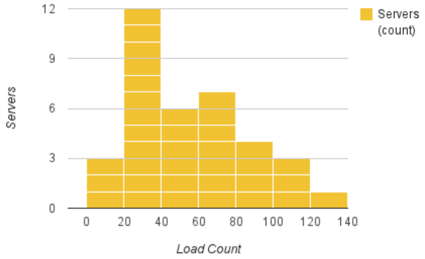

Distribution of Server Load (LA-256)

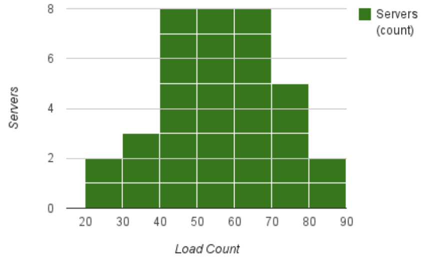

Distribution of Server Load (WQS-256)


Distribution of Server Load (LA-WQS-256)

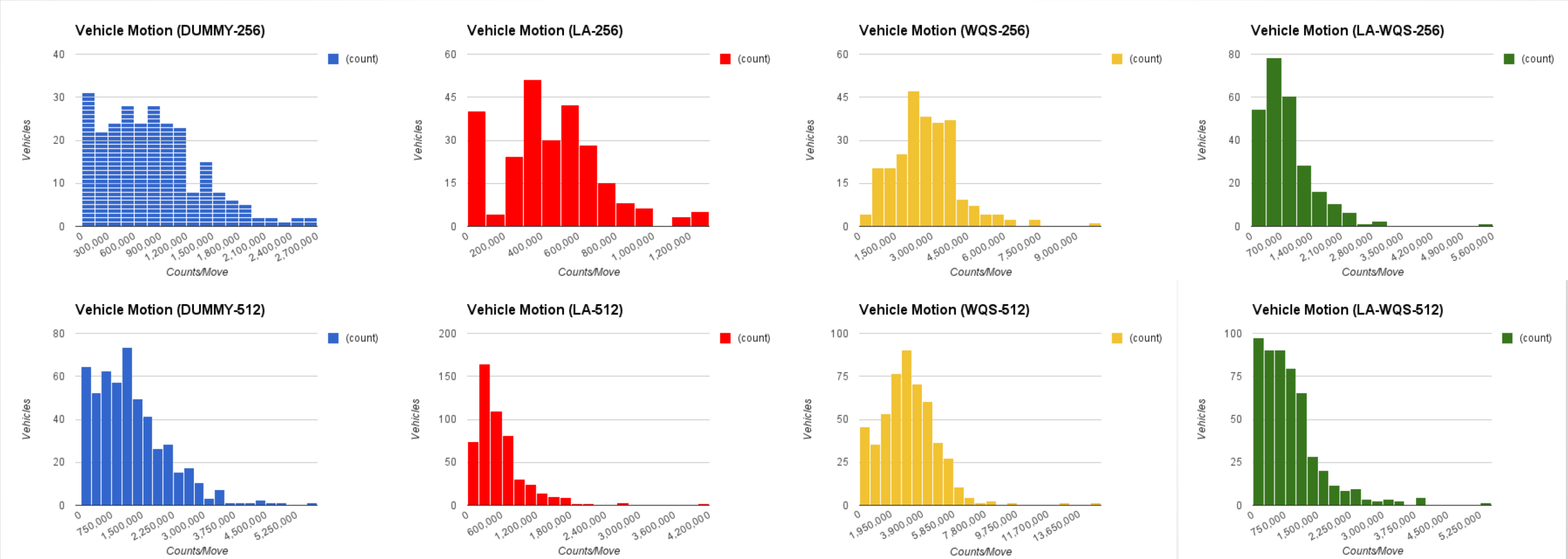

Distribution of Server Load (DUMMY-512)


Distribution of Server Load (LA-512)


Distribution of Server Load (WQS-512)


Distribution of Server Load (LA-WQS-512)

Vehicle Motion



Future Improvements

- Extending the look-ahead algorithm to search even further.
- Caching information
- Publisher/Subscriber model
- More complicated state models
- Simulate on real-world data

Conclusion

- Modeled a sophisticated problem into a simple framework
- Proposed two categories of algorithms for open research
- Successfully reduced travel time and achieved load-balancing on intersections
- Implemented a simulation tool for benchmarking

Fork us on github

https://github.com/cvhu/DistributedTrafficControl

Fork us on github

https://github.com/cvhu/DistributedTrafficControl

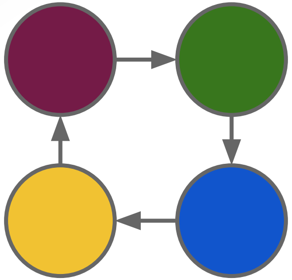
Back-up Slides

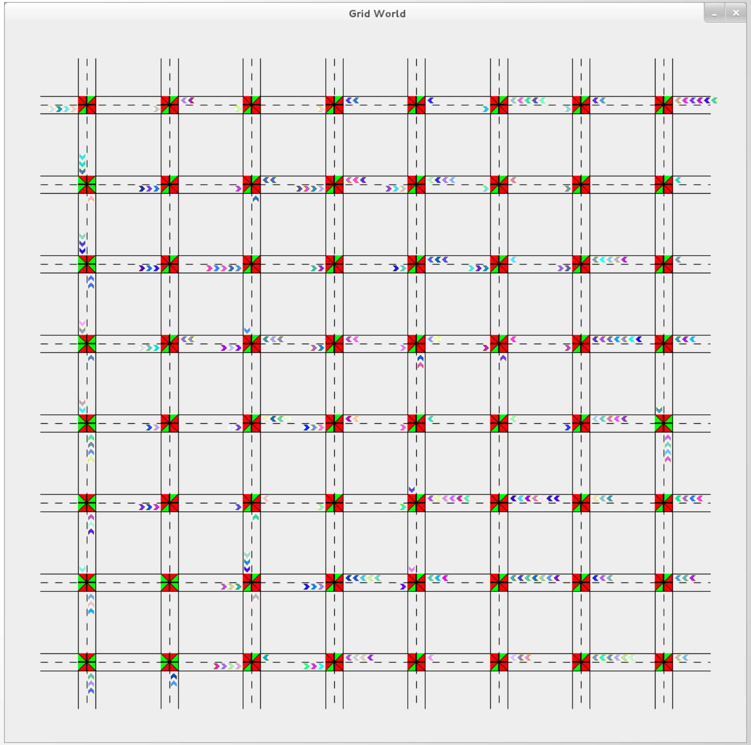
Dummy Algorithm

- Simple state machine
- No line communication between intersections (servers)
- No consideration given to state of lanes
- Issues
  - Naively steers traffic towards center of grid
- Benefits
  - Simplicity

Lookahead Algorithm

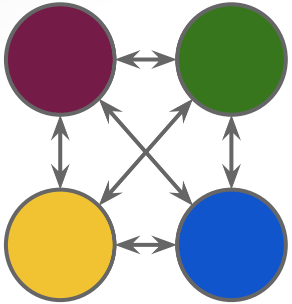
- Directs vehicles to less busy intersections
- Issues:
  - Greater message complexity
- Benefits
  - The number of vehicles per intersection is more evenly distributed

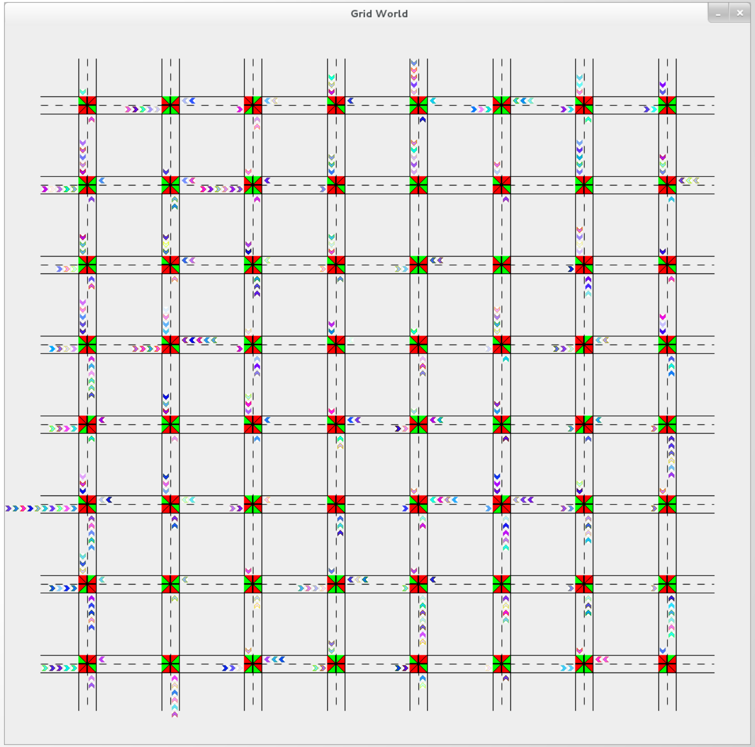




WeightedQ-Switcher Algorithm

- Next state serves “heaviest” lane
- Issues
  - Traffic naively directed towards the center of the grid
- Benefits
  - Greater throughput on each intersection





LA-WQS Algorithm

- Combination of lookahead and weightQ-switcher