COSC201

# Assignment 2

Jay Lee 3976743

## Introduction

This report covers the different implementations of some schemes for ‘packet management’ and also discusses the effectiveness of various strategies. The four implementations we cover in this report are the Basic Sensible Manager, Load-Aware Manager, Priority Manager and Diagonal Priority Manager. We will go into further detail about these managers later in the report.

### Manager Implementations

#### Basic Sensible Manager

The Basic Sensible Manager has packets stored in a first in, first out queue. There are no requirements to be stored in the queue meaning packets that are already in its destination location will still be stored in the queue.
The packets are then removed from the queue until one is found that needs to move to its destination. It moves towards its destination preferring to change row number if possible and then the column number.

#### Load-Aware Manager

The Load-Aware Manager has packets stored in a first in, first out queue. However, this queue will never store a packet that does not need to be moved.
The packets are then removed from this queue and if the packet’s destination is a straight line from its current location, it will just move one step closer in the appropriate direction each round. If the packet has two reasonable choices, it will check to see how many packets

are held at each of the locations and will be sent to the location with fewer packets. If the two locations have the same number of packets, it will prefer to change the row number and then the column number

### Priority Manager

The Priority manager has packets stored in a priority queue which never stores a packet that does not need to be moved. The priority of this packet is determined by the remaining total distance to its location along the grid i.e. by the sum of the absolute differences of the current and destination row and column numbers.
The packets with the highest priority are then removed from the priority queue (packet that needs to move the furthest). If its destination is in a straight line from its current location, it will move one step closer in the appropriate direction each round. If the packet has two reasonable choices, it will check to see how many packets are held at each of the locations and will be sent to the location with fewer packets. If the two locations have the same number of packets, it will prefer to change the row number and then the column number

### Diagonal Priority Manager

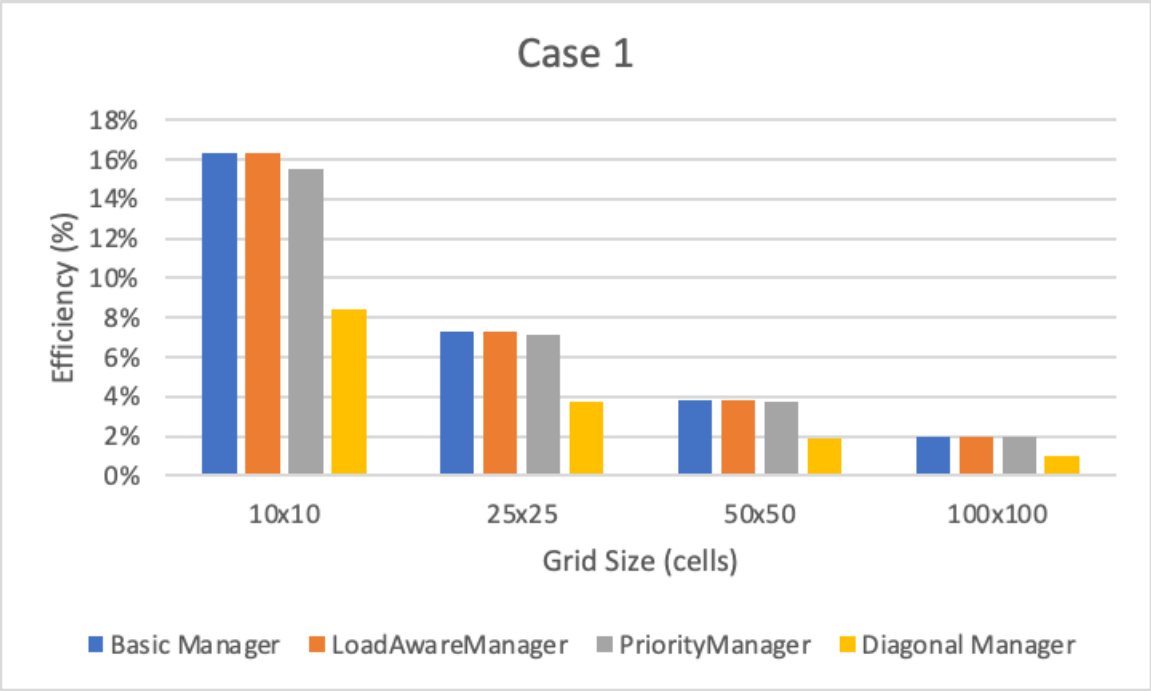
The Diagonal Priority manager has packets stored in a priority queue which never stores a packet that does not need to be moved. The priority of this packet is determined by the remaining total distance to its location along the grid i.e. by the sum of the absolute differences of the current and destination row and column numbers.
The packets with the highest priority are then removed from the priority queue (packet that needs to move the furthest). If its destination is in a straight line from its current location, it will move one step closer in the appropriate direction each round. However, if the packet is not in line with its destination, it will move diagonally one step in the appropriate direction.

## Cases

To help understand the effectiveness of these various implementations of 'PacketManager', we will explore different cases which have the warehouses set up in certain ways to see how they affect the efficiency of these different managers.
The efficiency of these managers is calculated by dividing the greatest distance a single packet had to travel by the total 'time' required to complete the given protocol. By time, we really mean the total number of rounds required to complete the protocol.
For each manager implementation, each test was repeated 100 times to then calculate the average efficiency of that implementation on various grid sizes.

### Case 1

This case required that the number of packets be equal to the number of cells in the grid. All the packets start in the upper left corner and have a different destination (in this scenario, different packets could not have the same destination)



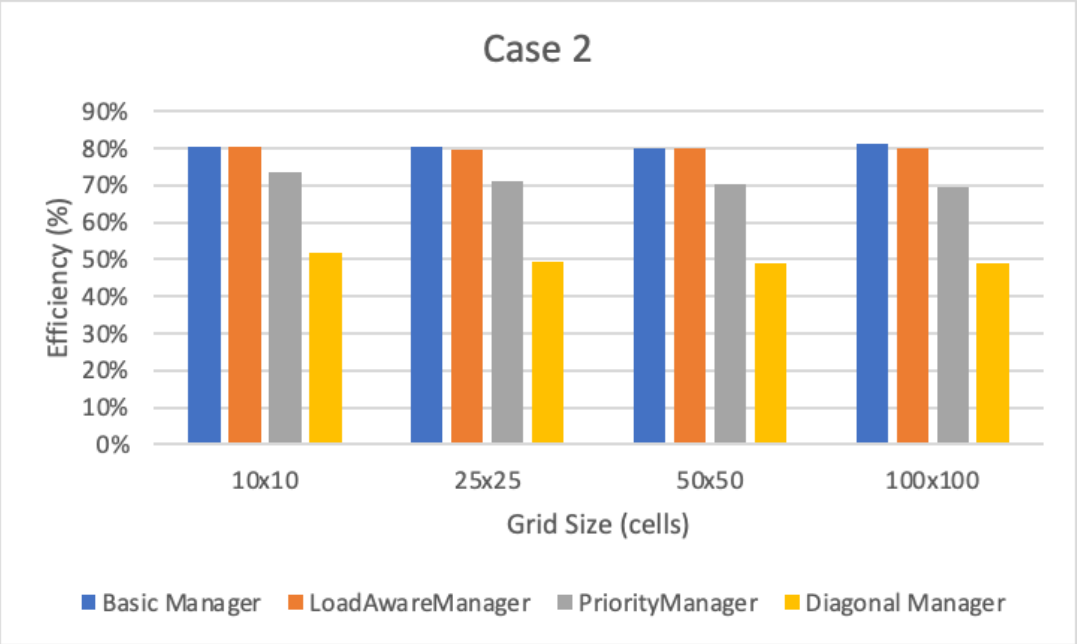
As can be seen in the graph above, the Basic Sensible, Load Aware and Priority Managers have very similar efficiencies to each other in all tested grid sizes. On the other hand, the Diagonal Priority Manager appears to have around half the efficiency as the other managers for all grid sizes.

As grid sizes increase, the efficiency of all the managers decreases which can be attributed to the fact that with more cells and packets, there is more distance for the packets to travel from their current location (the upper left cell) hence, the decrease in efficiency as grid size increases.

Despite the fact that the Priority Manager utilizes a priority queue, both the Basic Sensible and Load-Aware managers perform slightly better than the Priority Manager in terms of efficiency. This was unexpected as the Load-Aware and Priority Managers have the exact same sendPackets implementation and I originally hypothesised that the priority queue would make the Priority Manager more efficient.

## Case 2

This case required that the number of packets be equal to the number of cells in the grid. Each cell in the established grid has one packet to begin with and they all have a random destination (different packets may have the same destinations).



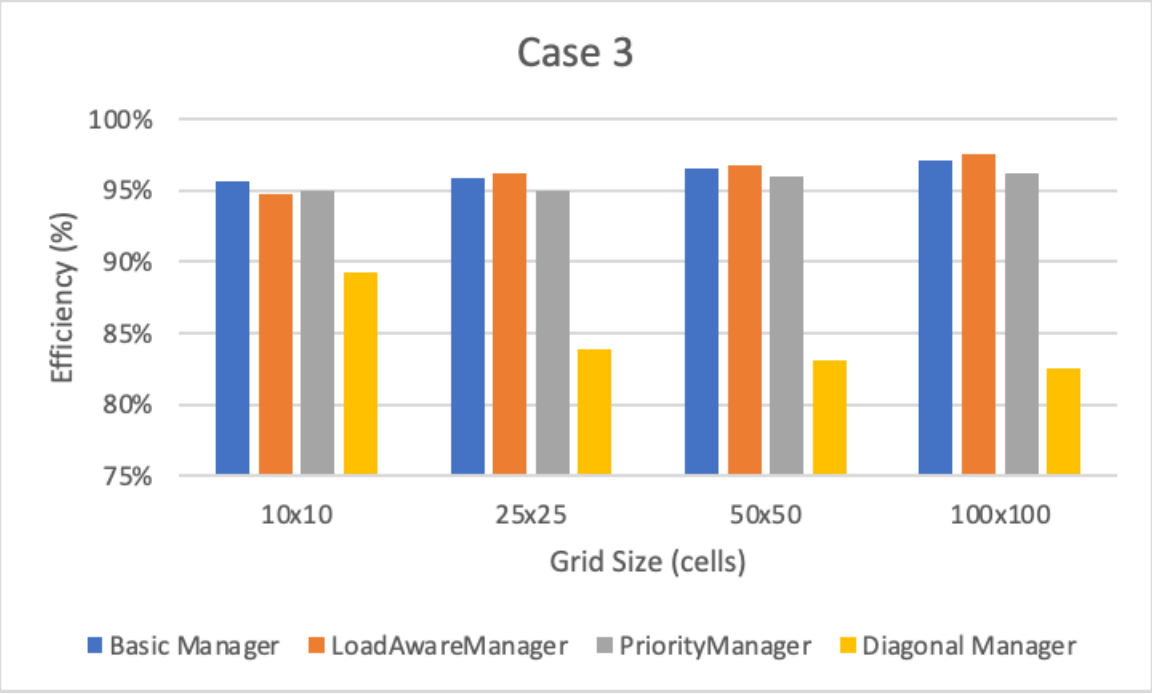
Similar to case 1, the Basic Sensible, Load Aware and Priority Managers have similar efficiencies to each other across the different grid sizes. Basic Sensible and Load Aware having ~1% difference in efficiency between them. The Priority manager is about ~10% less efficient than Basic Sensible and Load Aware and the Diagonal Priority Manager is about ~30% less efficient than Basic Sensible and Load Aware.

In contrast to case 1, the efficiencies of the different manager implementations show no significant variation across the different grid sizes which can be attributed to the fact that all the packets start off in different cells. This initial 'spread' of the packets across the entire cell means that an increase in grid size will have no significant effect on the distance a certain packet will have to travel in this case.

Once again, the Priority Manager performing worse in terms of efficiency compared to the Load Aware manager was unexpected as I thought the priority queue implementation would mean for more efficient sorting of the packets. However, this proved to not be the case.

### Case 3

This case is exactly the same as case 2 except each cell only has a 25% chance of containing a packet.

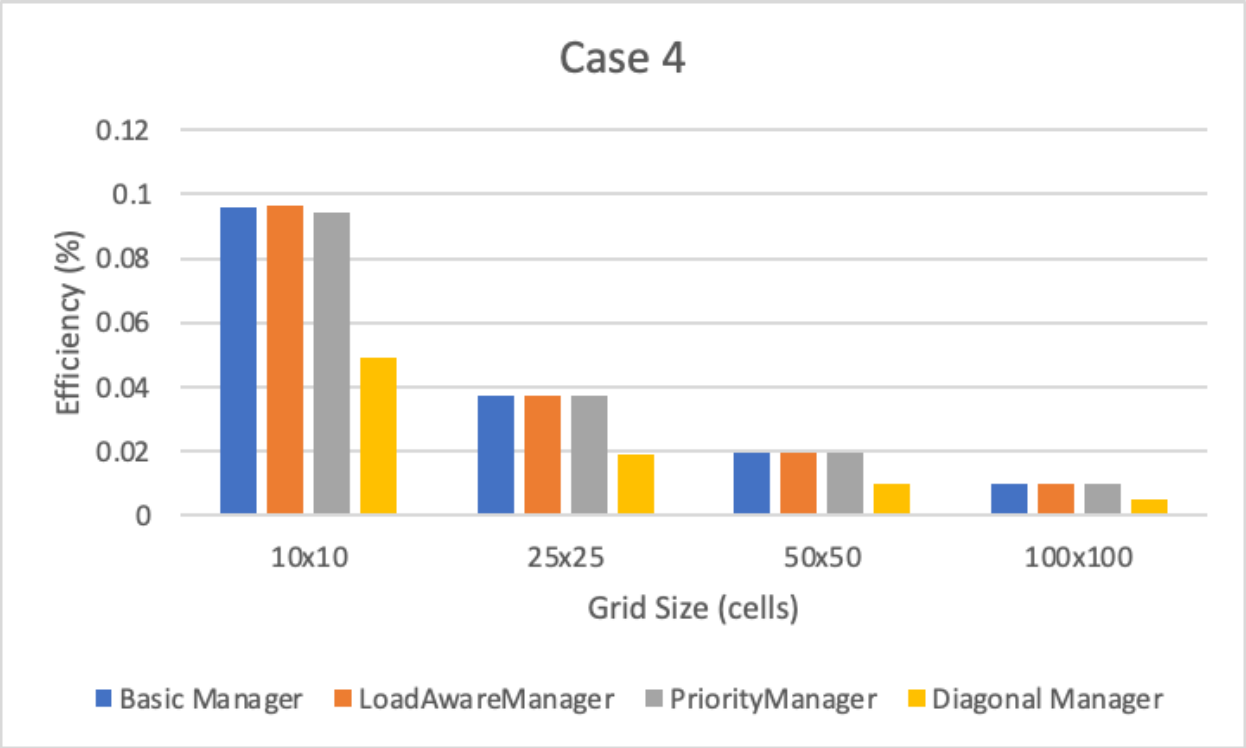


For the Basic Sensible, Load Aware and Priority Managers, the all performed very similarly across all the different grid sizes with their efficiencies floating around ~95%. On the other hand, the Diagonal Priority manager performed significantly worse in terms of efficiency compared to the other implementations.

For the grid sizes 25x25, 50x50 and 100x100 the Diagonal Priority Manager had a slightly decreasing average efficiency from around 84% to 82% as the grid sizes increased. However, in the 10x10 grid it had an average efficiency of around ~89%. This difference in efficiency is interesting seeing as the other manager implementations had slightly increasing average efficiencies as the grid sizes increased.

### Case 4

This case is exactly the same as case 1 except the starting location of all the packets is in the centre cell of the grid.

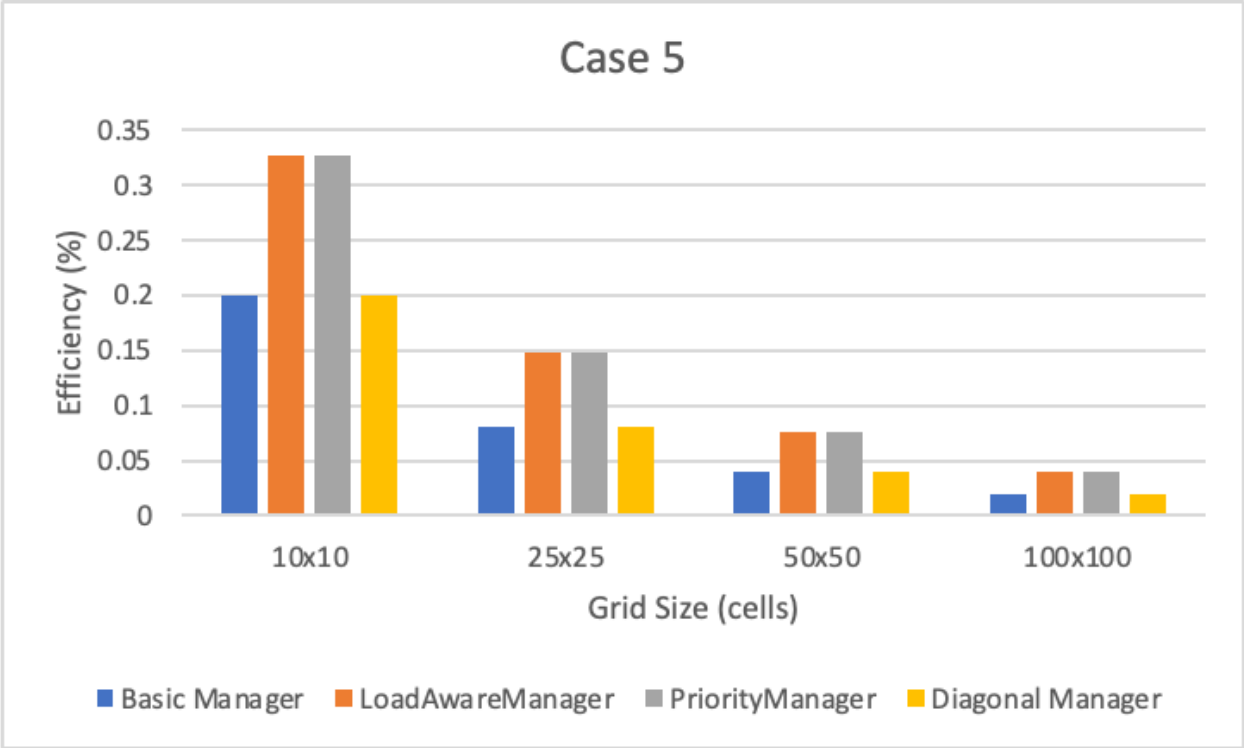


Overall, the trend observed in the graph is very similar to the one seen in case 1. This can be explained by the fact that as the grid size increases, the distance that some packets have to travel (if their destinations are on/or close to the edge cells) is drastically increased.

Although the overall trend is very similar, the actual average efficiency values are extremely different. The average efficiencies of the implementations in this case range from ~0.09% all the way down to ~0.005% compared to the range in case 1 (from ~16% down to ~1%).

### Case 4

In this case, the number of packets and cells are equal. The packets all start in random locations but they all have the same destination location.




Unlike all of the other cases, the Basic Sensible Manager actually performs the worst in terms of efficiency alongside the Diagonal Priority Manager. Seeing as both Load Aware and Priority Manager have very similar average efficiencies across all the grid sizes, we can claim that priority queue does not make a significant difference in the efficiency of packet delivery. However, we can infer that taking into account the loads of the surrounding cells does dramatically increase the efficiency of that packet manager implementation. This is

shown by how the Load Aware Manager and Priority Manager both are twice as efficient than the Basic Sensible and Diagonal Priority.

Similar to both case 4 and case 1, as grid size increases, the efficiencies of the different managers decrease due to the fact that an increase in grid size directly correlates to longer distances that packets have to travel (across the grid).

## Discussion

When comparing across all the cases, case 3 showed the highest average efficiencies across all methods which makes sense as there are far less packets than there are cells in the grid. This allows for less 'clutter' (multiple packets in the same cell) and means the packets are able to move around the grid more often in each round - allowing the packets to arrive at their destination quicker.

As mentioned above, I originally hypothesised that the implementation of the priority queue data structure would increase the efficiency of the manager that implemented it but the tests on the different cases showed no significant difference in efficiencies.

Overall, the Basic Sensible, Load Aware and Priority Manager performed very similarly across the different cases except for case 5 where Basic Sensible performed significantly worse in terms of efficiency due to not taking into consideration the loads of the surrounding cells when shifting packets.

Across test cases 1, 2, 3 and 4 the Diagonal Priority Manager performed the worst out of all of the implementations and in case 5 was very similar to the Basic Sensible Manager.

Given that the Diagonal Priority Manager allowed packets to be moved diagonally (which would usually take two moves for any of the other implementations), I expected it to perform better than it did. However, this did not turn out to be the case.

## Conclusion

Both the Load Aware and Priority Managers can be considered the most consistent and efficient manager implementations out of all of the managers in the test cases that were provided.