TRUCK PLATOONING

MODELLING AND IMPLEMENTATION

Team Alpha
Akhilesh Kakkayamkode
Arjun Veeramony
Angel Mary

Fachhochschule Dortmund
University of Applied Sciences and Arts

01

INTRODUCTION

02

MODELLING

03

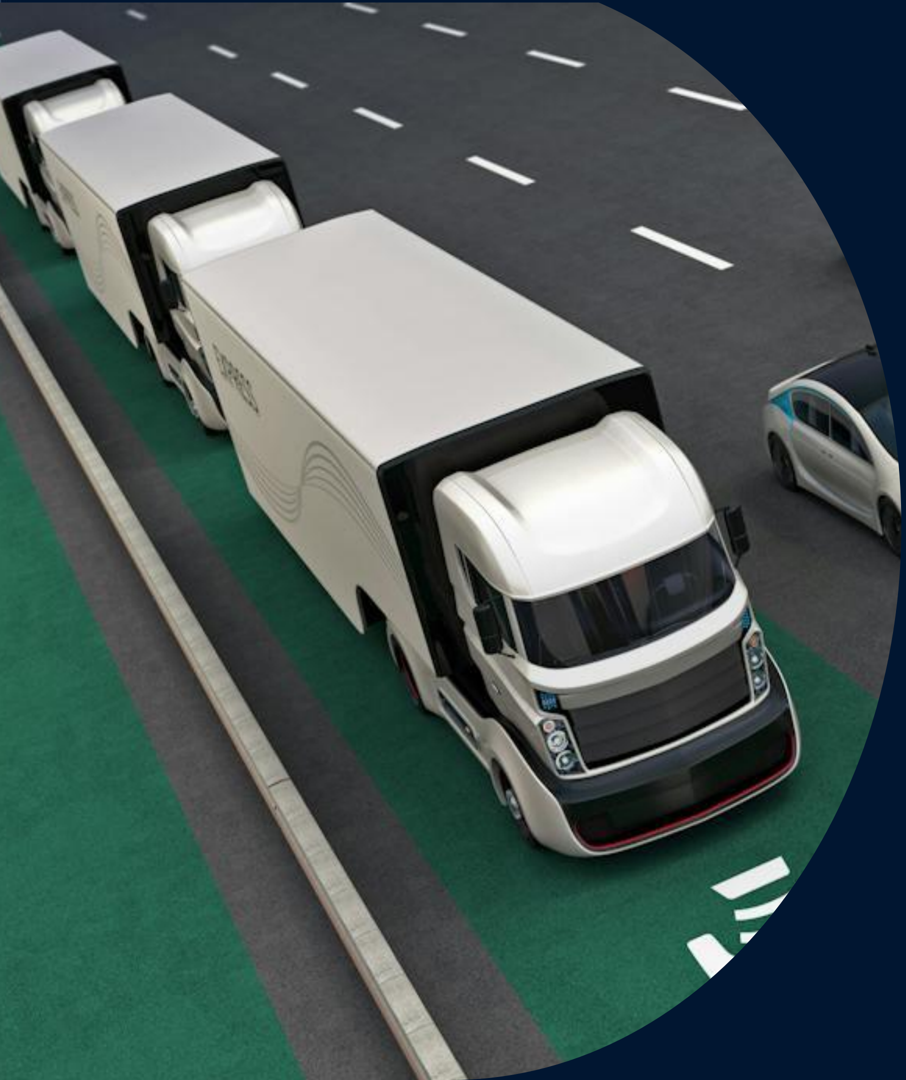
ARCHITECTURE

04

IMPLEMENTATION

01 INTRODUCTION

- Truck Platooning
- Signals and Data for Communication
- Maintaining Minimum Distance



TRUCK PLATOONING

- What is **Truck Platooning**?

  Two or more trucks linked to a leading truck.

- **Benefits** of Truck Platooning
  - Improved Fuel Economy
  - Reduced Carbon Emissions
  - Increased Safety
  - Reduced Traffic Jams
  - Less Manpower

DATA AND SIGNALS IN COMMUNICATION

PLATOON FORMATION SIGNALS

ACCELERATION/ DECELERATION DATA

CONNECTIVITY CHECK SIGNALS

LANE CHANGE ALERT

OBSTACLE DETECTION ALERT

EMERGENCY ALERT

BRAKING STATUS

TRAFFIC SIGNAL STATUS

Maintaining Minimum Distance

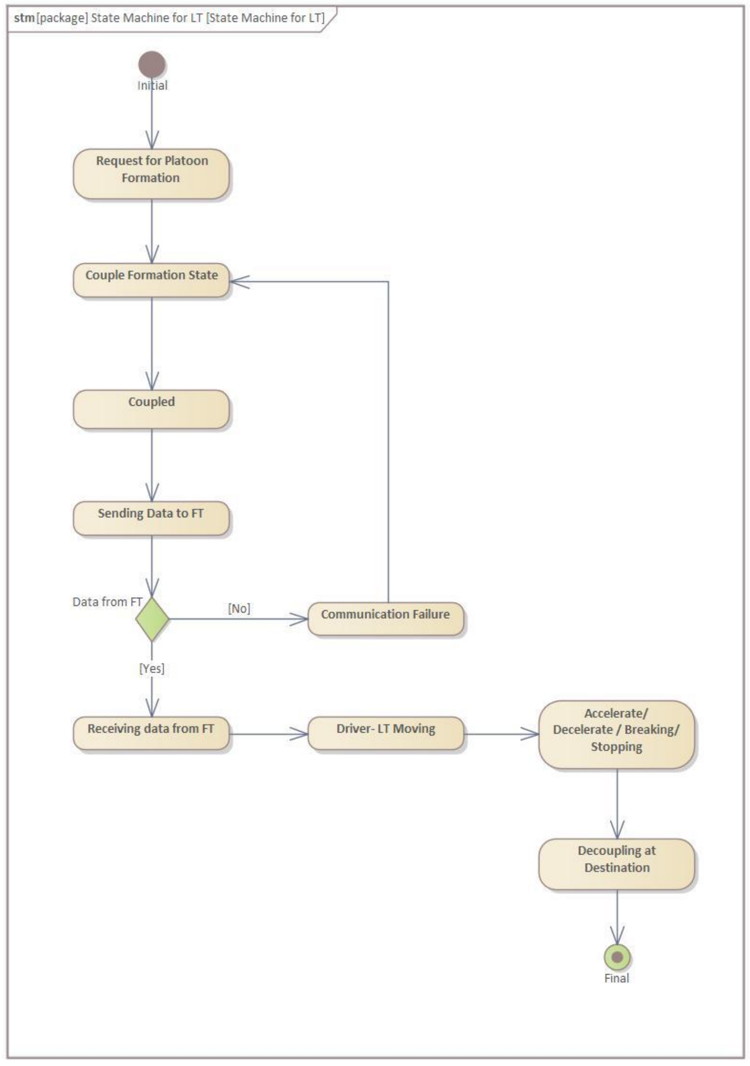
V2X Data (GPS Coordinates)

Radar and Camera Sensors

02

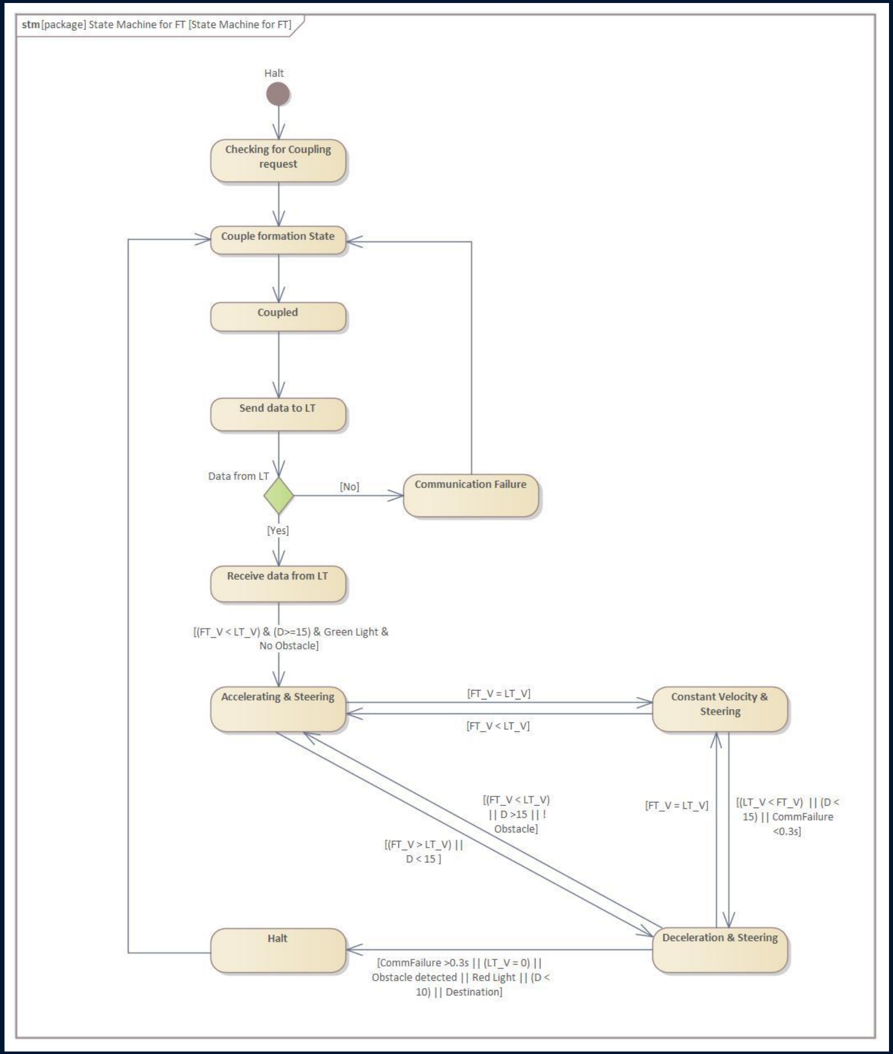
MODELLING

- State Machine Diagrams
- Activity Diagram
- Sequence Diagram



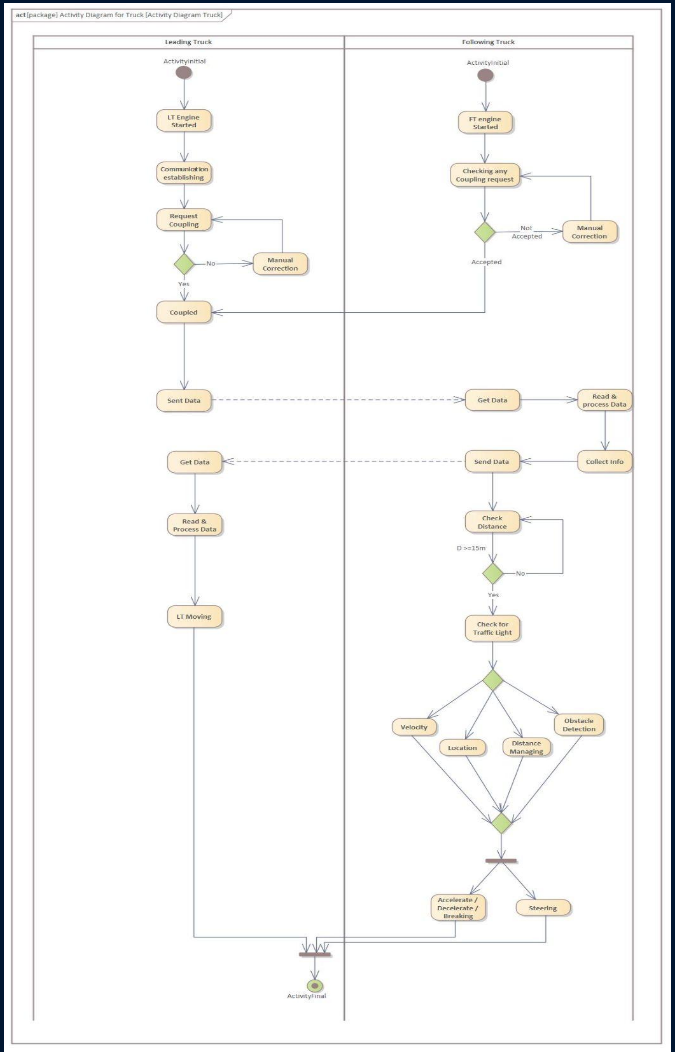
UML MODEL

State Machine Diagram
for Leading Truck



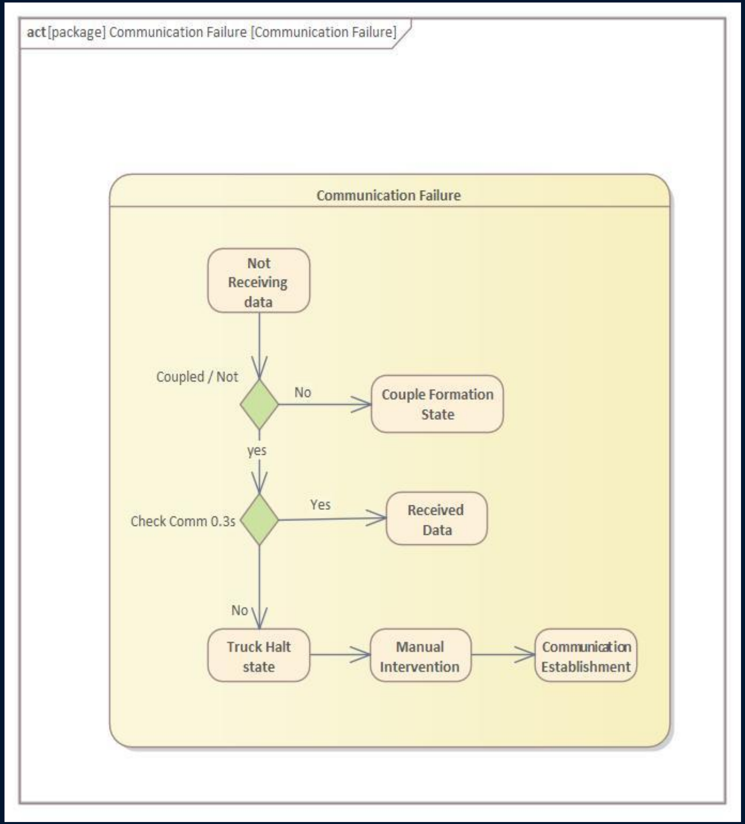
UML MODEL

State Machine Diagram for Following Trucks



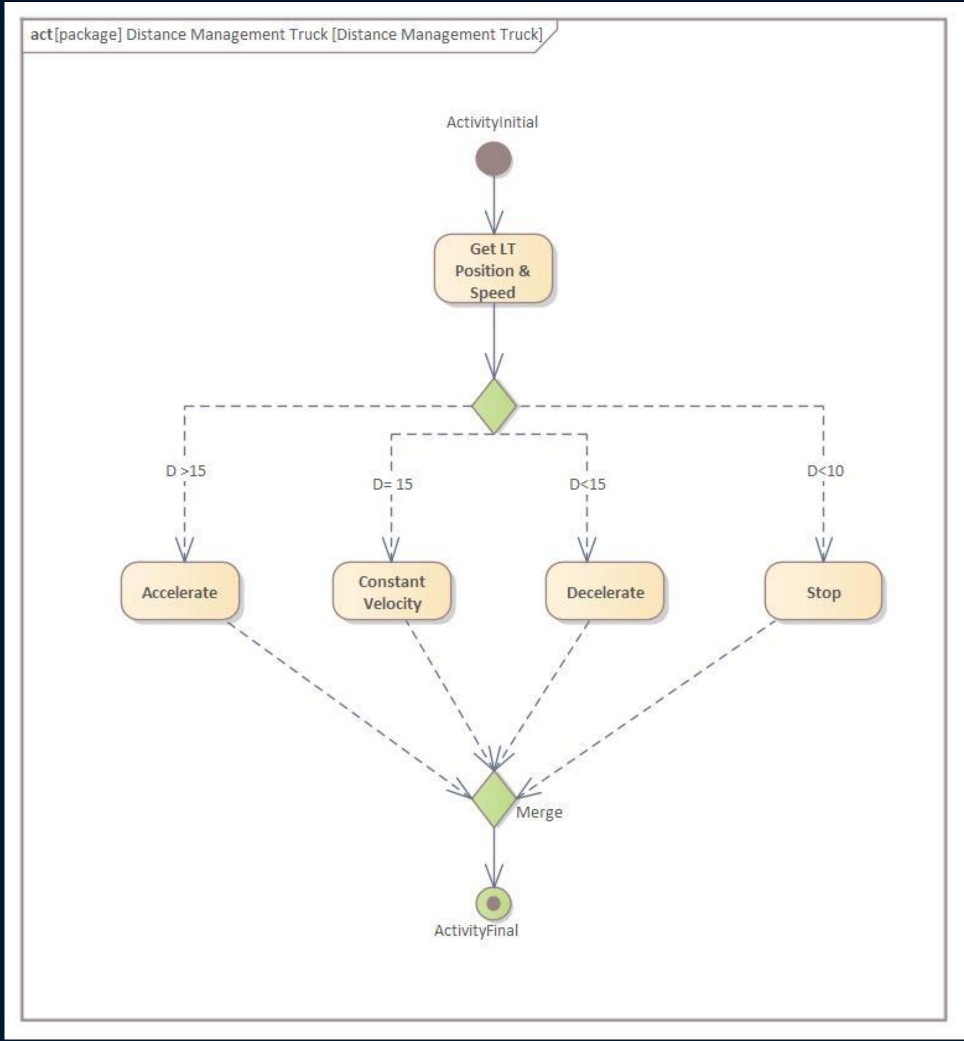
UML MODEL

Activity Diagram



UML MODEL

Communication Failure

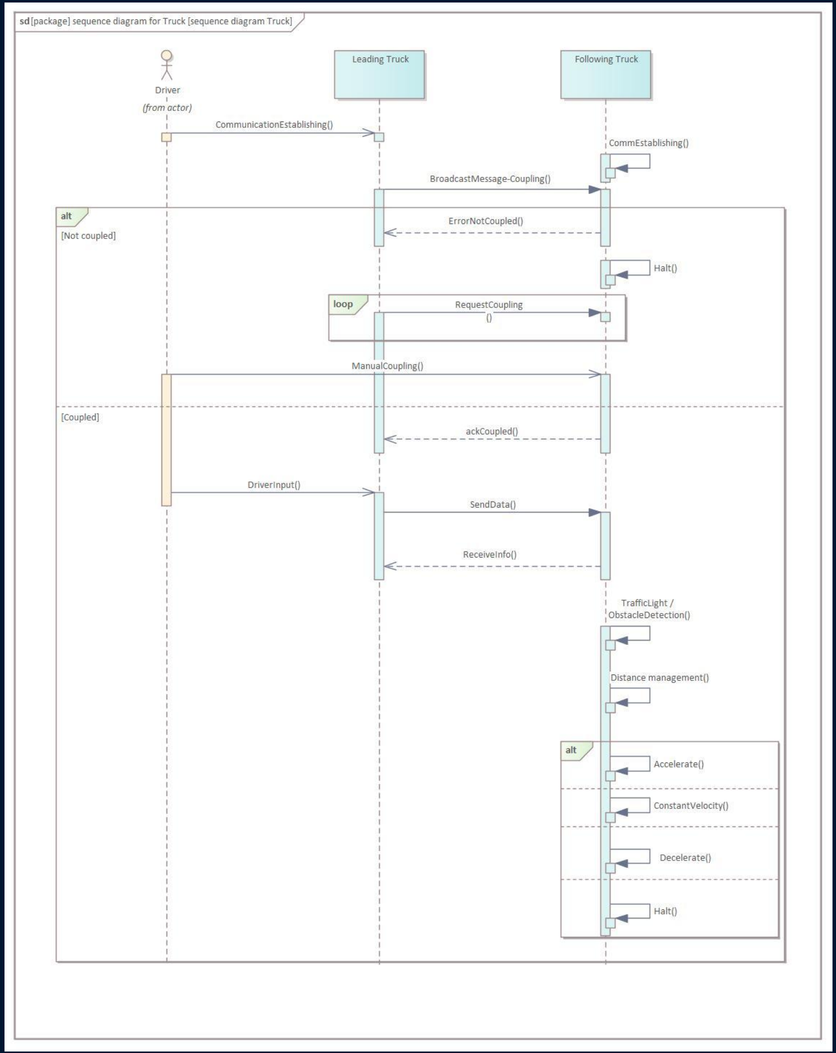


UML MODEL

Distance Management

UML MODEL

Sequence Diagram



03 ARCHITECTURE

- Parallel Programming Approaches
- Comparison
- Suggested Approach

PARALLEL PROGRAMMING APPROACHES

ASPECTS	MASTER SLAVE MODEL	NODE ONLY MODEL
CONTROL	Centralized Control with Master Coordinating Tasks	Decentralized Control with each node manages tasks independently
COMMUNICATION	Intensive Communication between Master and Slave	Limited Communication between Nodes
FAULT TOLERANCE	Critical mechanisms needed for Master Slave failures	Distributed Control enhances Fault tolerance

SUGGESTED APPROACH

MASTER SLAVE MODEL

REASONS

Communication Efficiency:

- Communication between the master and slave vehicles is typically more straightforward in a centralized model.
- The master can broadcast commands to all slave vehicles, reducing the complexity of communication protocols.

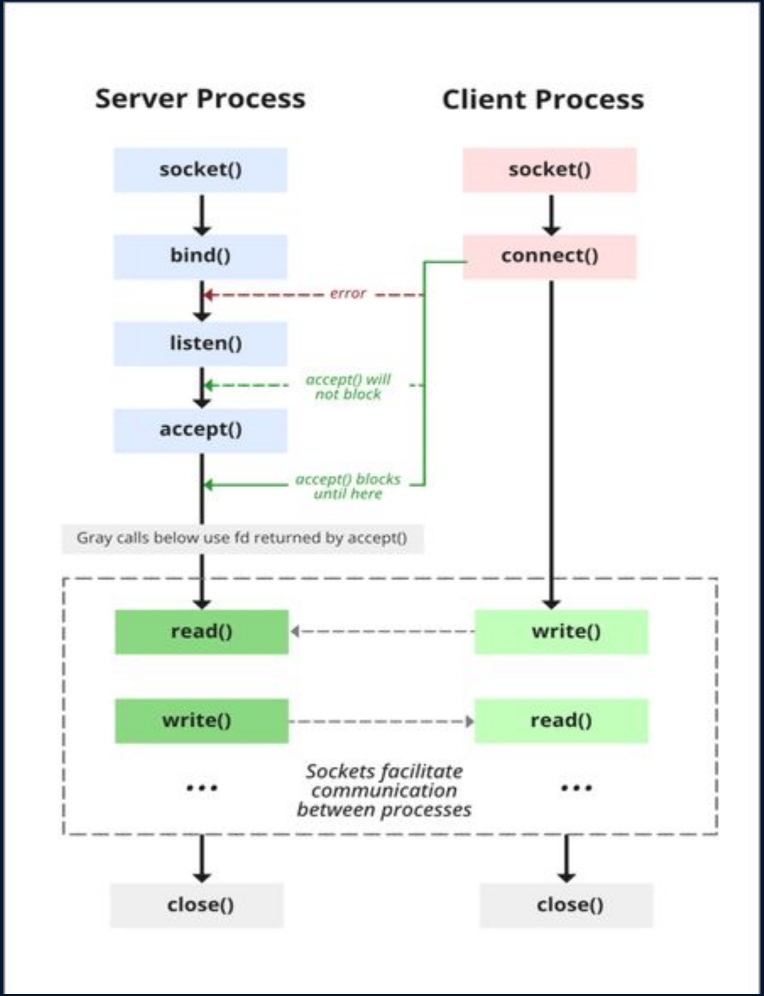
Adaptability:

- The master-slave model allows for dynamic adaptation to changing road conditions and traffic scenarios.
- The lead vehicle can adjust the platoon's behavior based on real-time data and make decisions that benefit the entire convoy.

04

IMPLEMENTATION

- Communication Protocol
- Working Approach
- GPU Implementation



Communication Protocol

TCP/IP SOCKET
PROGRAMMING

Working Approach

Server.cpp > main()

```cpp
#pragma omp section
        {
            std::string line;
            while (std::getline(inputFile, line))
            {
                if (line == "Accelerate")
                {
                    std::cout << "Parent Truck Accelerating:\n";
                    int t = 1;
                    //communicator(t, clientSocket);
                    myFIFOList.appendValue(1);
                    for (int i = 0; i < 5; ++i)
                    {
                        speed += 5;
                        std::cout << "Accelerating. Speed: " << speed << "\n";
                    }
                }
                else if (line == "Break")
```

```cpp
        }
        if(receivedNumber==5)
        {
            int DC=getRandomNumber();
            std::cout<< DC<<"Distance Check:\n";
            if(DC==1)
            {
                std::cout<<"Distance not optimal:\nReducing/Increasing Speed to maintai
                for (int i = 0; i < 2; ++i) {
                    std::cout << "adjusting speed" <<"\n";
                }
                std::cout<<"Optimal Distance between trucks are maintained.\n";
            }
            else{
                //std::cout<<"The distance between trucks is optimal.... "<<DC<<"\n";
            }

        }
    }
    int Obs=getRandomNumber();
      std::cout<< "Checking for Obstacle:\n";
       if(Obs==1)
       {
           std::cout<<"Obstacle detected:\nReducing Speed:\n";
           for (int i = 0; i < 2; ++i) {
               // Decrease speed by 5 for each iteration
               speed -= 2;
               std::cout << "Speed: " << speed << "\n";
```

Output

C:\Windows\System32\cmc

```
C:\Users\achua\OneDrive\Desktop\Truck>Server
Server listening on port 12345...
Server sent: Platoon Formation Request
Server received: Accepting request
Canged ttt
Parent Truck Accelerating:
Accelerating. Speed: 5
Accelerating. Speed: 10
Accelerating. Speed: 15
Accelerating. Speed: 20
Accelerating. Speed: 25
Parent Truck Accelerating:
Accelerating. Speed: 30
Accelerating. Speed: 35
Accelerating. Speed: 40
Accelerating. Speed: 45
Accelerating. Speed: 50
Parent Truck Accelerating:
Accelerating. Speed: 55
Accelerating. Speed: 60
Accelerating. Speed: 65
Accelerating. Speed: 70
Accelerating. Speed: 75
Parent moving in Constant Velocity
Speed: 75
Speed: 75
Speed: 75
Parent moving in Constant Velocity
Speed: 75
Speed: 75
Speed: 75
Turning
Updating Turning angles and Location data in Server
Parent Truck Breaking/deaccelerating:
Speed: 70
Speed: 65
Speed: 60
Speed: 55
Speed: 50
Parent Truck Breaking/deaccelerating:
Speed: 45
Speed: 40
Speed: 35
Speed: 30
Speed: 25
Parent Truck Breaking/deaccelerating:
Speed: 20
Speed: 15
Speed: 10
```

C:\Windows\System32\cmd.e

```
Journey Ended.....

C:\Users\achua\OneDrive\Desktop\Truck>Client
Client Received Platoon Formation Request
Client Accepting request
Client sent: Accepting request
----------------------------------------------------------------
Checking for Obstacle:
Child Truck Accelerating:
Accelerating. Speed: 5
Accelerating. Speed: 10
Accelerating. Speed: 15
Accelerating. Speed: 20
Accelerating. Speed: 25
----------------------------------------------------------------
Checking for Obstacle:
2Distance Check:
Checking for Obstacle:
Child Truck Accelerating:
Accelerating. Speed: 30
Accelerating. Speed: 35
Accelerating. Speed: 40
Accelerating. Speed: 45
Accelerating. Speed: 50
----------------------------------------------------------------
Checking for Obstacle:
2Distance Check:
Checking for Obstacle:
Obstacle detected:
Reducing Speed and increasing the distance:
Speed: 48
Speed: 46
Obstacle gone:
Increasing to speed to maintain optimal distance:
Child Truck Accelerating:
Accelerating. Speed: 55
Accelerating. Speed: 60
Accelerating. Speed: 65
Accelerating. Speed: 70
Accelerating. Speed: 75
----------------------------------------------------------------
Checking for Obstacle:
Obstacle detected:
Reducing Speed and increasing the distance:
Speed: 73
Speed: 71
Obstacle gone:
Increasing to speed to maintain optimal distance:
2Distance Check:
```

GPU IMPLEMENTATION

- **GRAPHICS CARD** - AMD RADEON 530
- **OPENCL** FRAMEWORK

```
ret = clGetDeviceIDs(platforms[0], CL_DEVICE_TYPE_GPU, 1,
    &device_id, &ret_num_devices);

// Create an OpenCL context
cl_context context = clCreateContext(NULL, 1, &device_id, NULL, NULL, &ret);

// Create a command queue
cl_command_queue command_queue = clCreateCommandQueue(context, device_id, 0, &ret);

// Create memory buffers on the device for each variable
cl_mem velocities_mem_obj = clCreateBuffer(context, CL_MEM_READ_ONLY,
    LIST_SIZE * sizeof(float), NULL, &ret);
cl_mem adjusted_speeds_mem_obj = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
    LIST_SIZE * sizeof(float), NULL, &ret);

// Copy the truck velocities to their respective memory buffer
ret = clEnqueueWriteBuffer(command_queue, velocities_mem_obj, CL_TRUE, 0,
    LIST_SIZE * sizeof(float), velocities, 0, NULL, NULL);

printf( _Format: "Before kernel execution\n");

// Create a program from the kernel source code
cl_program program = clCreateProgramWithSource(context, 1,
    (const char**)&source_str, (const size_t*)&source_size, &ret);

// Build the program
ret = clBuildProgram(program, 1, &device_id, NULL, NULL, NULL);

printf( _Format: "After building\n");

// Create the OpenCL kernel
cl_kernel kernel = clCreateKernel(program, "platooning_communication", &ret);

// Set the arguments of the kernel
ret = clSetKernelArg(kernel, 0, sizeof(cl_mem), (void*)&velocities_mem_obj);
ret = clSetKernelArg(kernel, 1, sizeof(cl_mem), (void*)&adjusted_speeds_mem_obj);
ret = clSetKernelArg(kernel, 2, sizeof(float), (void*)&communication_gain);
```

GPU IMPLEMENTATION

HOST CODE

GPU IMPLEMENTATION

KERNEL CODE

```
// adj_speed_kernel.cl

__kernel void platooning_communication(__global const float* velocities, __global float* adjusted_speeds, float communication_gain) {
    int i = get_global_id(0);

    float current_velocity = velocities[i];

    // Simulate communication: adjust speed based on the average velocity of the platoon
    float sum_velocities = current_velocity;
    for (int j = 0; j < get_global_size(0); j++) {
        if (j != i) {
            sum_velocities += velocities[j];
        }
    }

    float average_velocity = sum_velocities / get_global_size(0);

    // Adjust the speed based on communication
    adjusted_speeds[i] = current_velocity + communication_gain * (average_velocity - current_velocity);
}
```

GPU IMPLEMENTATION

OUTPUT

Microsoft Visual Studio Debu

```
Truck Platooning Simulation
Kernel loading done
Before kernel execution
After building
After kernel execution
Truck Velocities:
30.00   35.00   28.00   32.00
Adjusted Speeds:
30.12   34.62   28.33   31.92
C:\Users\vkakh\Downloads\ConsoleApplication1\x64\D
Press any key to close this window . . .
```

REFERENCE

- Truck platooning application - S. Ellwanger, E. Wohlfarth, 2017 IEEE Intelligent Vehicles Symposium (IV)
- “An OpenMP Application Programming Interface, Examples”,Version 4.5.0, 11.2016.

THANK YOU!!