ABSTRACT

JAIN, ASHISH. Protocol Design for a Public Logistics Network. (Under the direction of Dr. Michael G. Kay.)

A public logistics network (PLN) is envisioned to be a multi-firm distribution system where several companies work in coordination for package transport such that packages maximize their values and trucks their profit. The presence of a multi-firm environment makes it difficult to have centralized control over the system and hence the coordination among various parts of the system becomes difficult as compared to a private logistics network where a single company controls the entire system. As a result, a coordination mechanism is required such that system work on its own without any central authority making plans for the operation of the PLN.

An analytical formula was developed to study the performance of a PLN. In this thesis, a simulation model of a PLN was developed and is used to determine the average waiting time for packages transported through a PLN for use in the analytical formula.

A set of protocols was proposed for the trucks and packages in a PLN to enable packages that value transportation the highest to be transported by trucks that can most efficiently transport the packages. In this thesis, the set of protocols was implemented in a simulation model and the performance of the PLN was compared with and without the protocols. The performance measures used were weighted waiting time and weighted transportation time for packages transported through a PLN. It was found that there was a significant decrease in waiting time associated with the use of the protocols.

Protocol Design for a Public Logistics Network

By

ASHISH JAIN

A thesis submitted to the graduate faculty of

North Carolina State University

In partial fulfillment of the

Requirements for the degree of

Master of Science

INDUSTRIAL ENGINEERING

Raleigh, NC

December, 2004

APPROVED BY

_____	_____
Dr. Russell E. King	Dr. Jennifer Blackhurst

_____

Dr. Michael G. Kay
(Chair of Advisory Committee)

BIOGRAPHY

Ashish Jain, born on the 1st of June, 1978 in the city of Khargone in the state of Madhya Pradesh, India, completed his schooling from the city of Jabalpur, India. He obtained his Bachelor of Technology (Hon.) degree in Mechanical Engineering from IIT Kharagpur.

During his graduate studies at North Carolina State University, he undertook course work in Industrial Engineering, Statistics and Supply Chain Management and worked in the area of discrete event simulation and designing a logistics system under the supervision of Dr. Michael G. Kay. During his graduate studies he did an internship and co-op at Philip Morris, USA. His research interests include logistics, supply chain and discrete event simulation.

ACKNOWLEDGEMENTS

I would like to thank the following people for their generous contributions to the completion of this work:

- My parents Mr. Rajiv Kumar Jain and Mrs. Sarvesh Jain, my brother Mr. Rishi Jain, sister-in-law Mrs. Smita Jain, my fiancée Ms. Nripa Tewari and my sister Ms. Anindita Banerjee for your love and support.
- Dr. Michael Kay as my committee chair and advisor, you have provided continuous support and guidance throughout the project. We have seen many ups and downs in this research work, but we have surely come a long way since the beginning of this research work. I also gained a lot of experience in handling a research project by working with you.
- My committee members Dr. Russell King and Dr. Jennifer Blackhurst for their encouragement and feedback.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES viii

1. Introduction 1

1.1 Introduction 1

1.2 Operations of a PLN 2

2. Estimating Package Waiting Time 10

2.1 Simulation Model 10

2.1.1 Overview 11

2.1.2 Input Parameters 12

2.1.3 Assumptions 14

2.1.4 Random Number Generation 14

2.1.5 Execution of simulation 15

2.2 Analytical Formula 18

2.2.1 Verification 24

2.3 Waiting Time from simulation versus Headway 25

3. Protocol Design 27

3.1 Basic Definitions 28

3.2 Package Protocol 30

3.2.1 Load Formation 30

3.2.2 Allocation of Load Bid 31

3.3 Truck Protocol 32

3.3.1 Priority for Accepting Loads 32

3.3.2 Loads at DCs along the Intended path 33

3.3.3 Intended Path 35
3.3.4 Reneging 38
3.4 Framework for Agent Co-ordination 38
3.5 Simulation 39
3.5.1 Assumptions 43
4. Results and Analysis 45
4.1 Overview 45
4.2 Input Data 47
4.2.1 Mean for Beta Distribution for Package Values 47
4.3 Results 48
5. Future Work 51
References 52
Appendix 1a 53
Appendix 1b 59
Appendix 1c 85

LIST OF FIGURES

Figure1.1: Packages P1, P2 and P3 coming from different DCs to DC7 and have different routes from DC8 have a link common in their routes i.e. from DC7 to DC8 2

Figure1.2: Hypothetical Network of a PLN [1] 3

Figure1.3: A package transported from DC8 to DC1 4

Figure1.4: A PLN with Distribution Centers, Pick-up Centers and Delivery Points 5

Figure1.5: Internet system showing Routers and local networks 5

Figure2.1: Flowchart showing various stages of package transportation in the simulation model 13

Figure2.2: Flowchart showing activities at the event of generation of demand 15

Figure2.3: Flowchart showing the activities at the event of package ready to be picked at the supplier DC 16

Figure2.4: Flowchart showing the activities at the event of truck arrival at a DC 17

Figure2.5: Flowchart showing activities at the event of truck departure 18

Figure2.6: Plot of Average Link Waiting Time Vs Net Link Demand for demand of 50000 packages per day, truck capacity 20 and load factor 0.7 21

Figure2.7: Exponential Equation fitted on the plot between ‘d’ and load factor 23

Figure3.1: Bid amount for package P1 from DC1 to DC2 and from DC2 to DC3 28

Figure3.2: Load formation at DC3 29

Figure3.3: Load formation at DC4 at time t = 0, 1 and 2. Trucks T1 and T2 will reach DC4 at time t = 1 and t = 2 respectively 30

Figure3.4: Priority to accept and reject load at DC3 33

Figure3.5: Before accepting load at DC1, truck needs to accept load at DC5 and DC8 ... 34

Figure3.6: Initially, load bids at DC5, DC3 and DC1 are low enough for truck T1; assuming truck requires $7 to travel between any two DCs. After $15, truck T1 accepts the bids at DC1 because it provides money for other links.....34
Figure3.7: Different routes possible for T1 to reach DC1 35
Figure3.8: Trucks end up getting less money because of shortest route constraint.....36
Figure3.9: Framework for the agent based co-ordination.....39
Figure3.10: Package arrival event in the simulation with protocol 40
Figure3.12: Event of truck leaving a DC in the simulation with protocol.....41
Figure3.11: Bidding Process.....42
Figure3.13: Event of truck arriving at a DC in the simulation with protocol.....44
Figure4.1: Packages at DC3 waiting to be transported to their respective next DCs 45

LIST OF TABLES

Table1.1: Costs for Transport of a Single Package from DC8 to DC1 4

Table1.2: Comparison between a PLN and the Internet 6

Table2.1: Set of values for simulation runs 19

Table2.2: Average Waiting Time and Average Transportation Time (in hours) for demand of 50000packages per day, truck capacity 20 and load factor 0.7 19

Table2.3: Average Waiting Time and Average Transportation Time (in hours) for demand of 70000packages per day, truck capacity 100 and load factor 0.8 20

Table2.4: Shows relation between parameters 'c' and 'd' for different sets of demand, truck capacity and load factors 22

Table2.5: R-square values for different equations on figure 2.7 22

Table2.6: Verification of the Equation (4) by assigning demand randomly among DCs.. 25

Table2.7: Verification of the Equation (4) by reassigning distances between DCs and also demand among DCs randomly 25

Table2.8: Comparison of "Waiting Time from simulation" with Half Headway and Full Headway 26

Table3.1: Allocation of load bid. 32

Table3.2: Different possible routes for truck T2 to reach DC1 from DC5 and time taken to reach DC1 37

Table3.3: Different possible routes for truck T2 to reach DC1 from DC5 and money it would receive from them 37

Table3.4: Complete route, money offered by packages on each route, transportation cost and net amount received by truck T2 37

Table4.1: Packages in Q1 at DC3 and their time of arrival (in the increasing order) and amount offered 46

Table4.2: Packages in Q2 at DC3 and their time of arrival (in the increasing order) and amount offered 46

Table4.3: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time 48

Table4.4: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time 48

Table4.5: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time 49

Table4.6: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time 49

Table4.7: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time. 50

Table4.8: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time 50

1. Introduction

1.1 Introduction

A public logistics network (PLN) has been proposed to provide fast, flexible and low-cost means of parcel transport [1]. The primary objective of a PLN is to build a logistics system where even though multiple companies could own different parts of the network, it functions in a cooperative manner to provide transportation services. For example, one company could own one or more distribution centers (DCs), another could own one or more trucks and yet another company could own a combination of trucks and DCs. This would allow even small companies with limited investment capabilities to participate actively in parcel transportation unlike a private logistics network like UPS or FedEx, which requires a huge investment by a single company and the same company controls all logistics operations. Nevertheless, a single company, unless it becomes a monopoly, is ultimately limited in the scale of its operation, resulting in the use of a limited number of large-scale hub transshipment points that can result in packages making many circuitous hops before reaching their destinations. In public logistics network, scale economies could be realized by establishing information systems that would facilitate providing the entire network's demand to each element of the system. This increase in scale might make it economical to ship full truckloads as opposed to costlier less-than-truckloads. This could be possible because a single truck load could be used to transport packages (associated with demand from different DCs) over a link, if that link is common to their routes. Figure 1.1 shows how packages at DC7, coming from different DCs and having different destinations, could form a single truckload to move from DC7 to DC8 because they have this common link in their routes. Package P1 coming from DC1 and going to DC4, P2 coming from DC2 and going to DC5, and P3 coming from DC3 and going to DC6. All have a common link from DC7 to DC8 and, hence, they could form a single

truckload, depending on their time of arrival at DC7. This can result in many of the long-haul single-product full-truckload shipments between private facilities being replaced by a series of short-distance hops between public DCs.

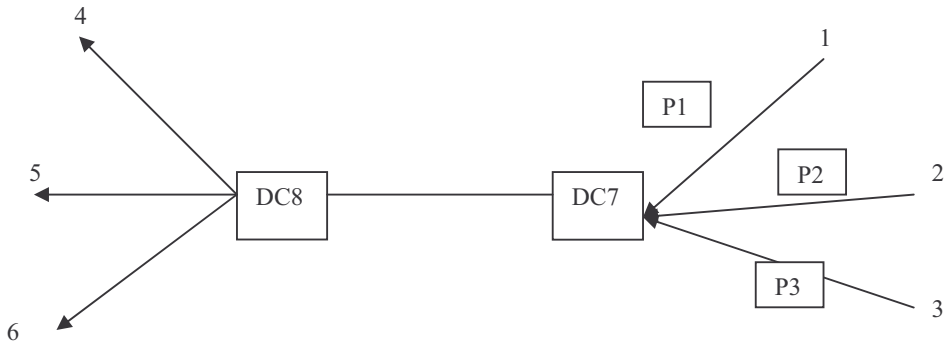


Figure1.1: Packages P1, P2 and P3 coming from different DCs to DC7 and have different routes from DC8 have a link common in their routes i.e. from DC7 to DC8

The presence of multiple firms in a PLN would make it impossible for a centralized control of the system as different companies would have different objective functions. Because of the decentralized nature of the system, a coordination mechanism needs to be developed which would be followed by each element of the system to work in a collaborative fashion. Kay [4] describes the protocols developed to provide coordination among various entities in a PLN.

1.2 Operations of a PLN

Figure 1.2 is a hypothetical public logistics network showing 36 DCs covering the southeastern portion of the USA and connected via interstate highways [1]. Each of the interstate DCs would serve as a hub in a sub-network of local DCs (not shown) covering the region surrounding the interstate DCs.

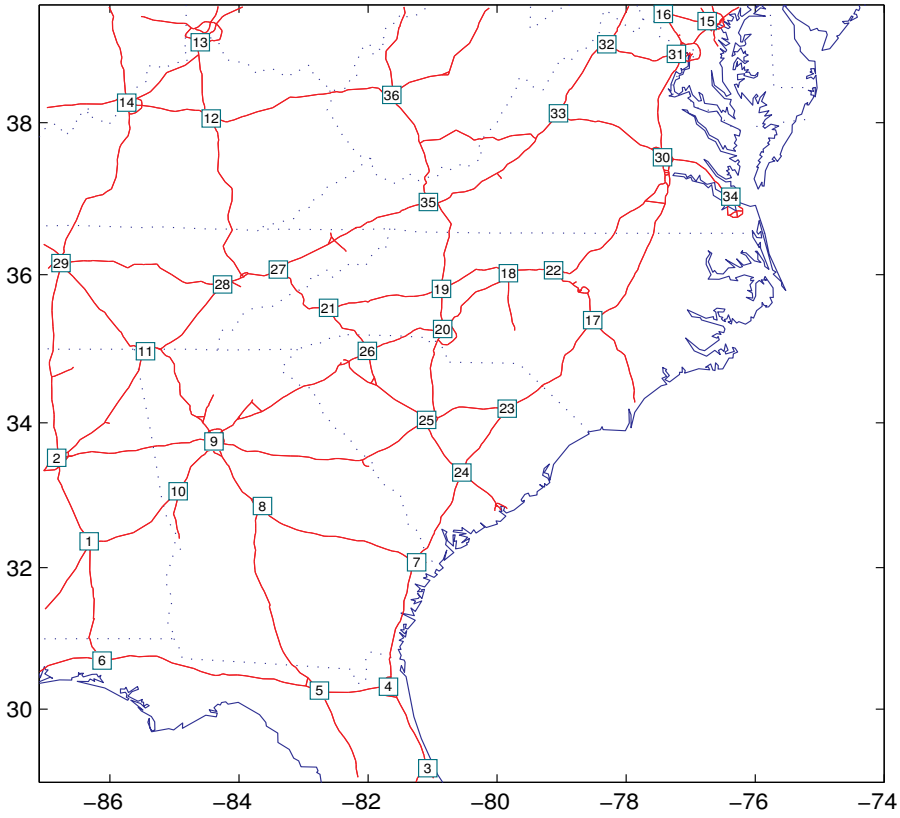


Figure1.2: Hypothetical Network of a PLN [1]

The decentralized control of a PLN and transportation of packages through it is analogous to the Internet and transmission of data packets through it. This section describes the analogy between PLN and the Internet. Consider an example as shown in Figure 1.3.

Figure 1.3 shows the transport of a single package from its pickup at DC8 through to its delivery at DC 1. The package travels on one truck from DC 8 to DC 5. At DC 5, it is unloaded, sorted, and then loaded onto what would usually be a different truck for its transport to DC 1. As listed in Table 1.1, the total time required for the transport of the package is the sum of (1) the loading/unloading (L/U) times at each DC, (2) the time spent waiting at each DC for arrival of a truck to transport the package to its next DC, and (3) the time spent in transit onboard each truck.

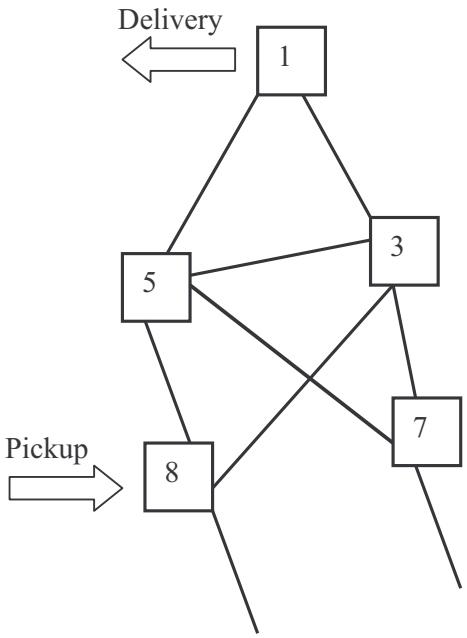


Figure1.3: A package transported from DC8 to DC1

Table1.1: Costs for Transport of a Single Package from DC8 to DC1

Activity No.	Activity	Time(min)	Credit ($)	Bid	Cost ($)
1	Pickup				
2	Unload at DC8	5	5.00		0.02
3	Wait at DC8	30	4.98		0.30
4	Load	5	4.68		0.02
5	Linehaul to DC5	60	4.66	2.00	2.00
6	Unload	5	2.66		0.02
7	Wait at DC4	20	2.64		0.20
8	Load	5	2.44		0.02
9	Linehaul to DC1	90	2.42	2.30	2.30
10	Unload	5	0.12		0.02
11	Delivery		0.10		
Total		225 min 3.75 hr		4.30	4.90

Figure 1.4 and Figure 1.5 show the high-level operations of a PLN and the Internet respectively. In Figure 1.4, there are four DCs viz. DC1, DC2, DC3 and DC4. “P” indicates pick-up centers and “D” indicates delivery points. A package moving from a pick-up center

near DC1 to a delivery point near DC3 goes to DC1 first and then through the shortest route (either through DC2 or DC4) reaches DC3, where it is delivered to one of the delivery points.

In Figure 1.5, there are four routers viz. R1, R2 R3 and R4. “C” represents a computer on a local network. Any data packet starting from a computer in the local network near R1 first goes to router R1 and then through the shortest route (either through R2 or R4) reaches R3, where it is delivered to one of the computers.

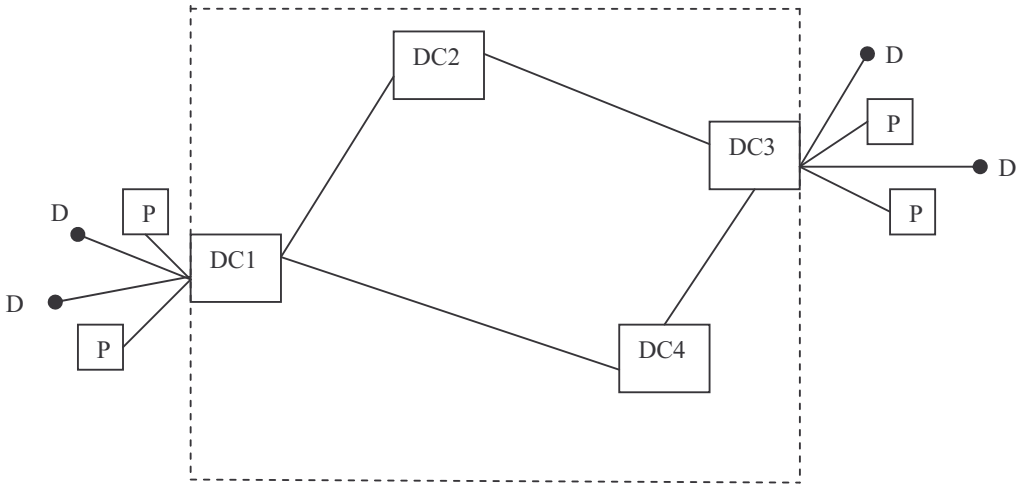


Figure1.4: A PLN with Distribution Centers, Pick-up Centers and Delivery Points

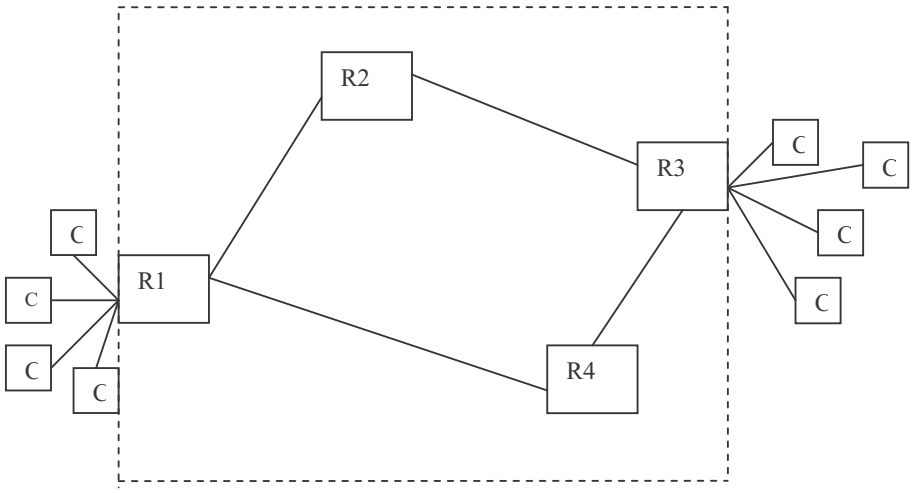


Figure1.5: Internet system showing Routers and local networks

For this thesis, only the portion inside the dotted lines in Figure 1.3 and Figure 1.4 is considered. Based on this consideration, Table 1.2 shows the analogy between a PLN and the Internet

Table1.2: Comparison between a PLN and the Internet

PLN	Internet
Distribution Centers	Routers
Physical Packages	Data Packets
Trucks	Wires/Cables

PLN is envisioned to be a flexible system which would help packages to maximize their values from transport and trucks their profits. Following are the features of PLN which describes the flexibility given to the entities of the system which would help them achieve their objectives:

1. In order for a truck to maximize its profit, it is free to make decisions, based on the choices given to it and the configuration of the system, about the routes it would take and hence the DCs it would go to pickup packages. The choices given to the truck would be based on the protocols that have been established to enable smooth operation. After a truck is ready to move on a route based on the money the truck would get, it is free to change its route at any point in time if it finds that some other route is offering it more money.
2. In order for a package to maximize its value from transport, a package is free to make decisions, based on the choices given to it and the configuration of the system, about the money it is willing to give for transportation, the trucks it would take for its journey, and the routes it would choose to reach its destination.

This is unlike a scheduling system where all trucks and packages follow a fixed schedule.

The flexibility of the system, mentioned above, results in the following issues:

1. Trucks would go where they could make maximum profit. This introduces uncertainty in the routes of trucks and the time at which they would arrive at DCs.
2. Package would like to be transported in such a way that it cost the least possible amount. Cost could be in terms of dollars or time depending on the objective function of the package. This could introduce uncertainty in time at which they would reach the destination and the transportation cost for the package.
3. Decisions have to be made at various stages of the journey, which requires continuous monitoring of the system by the entities themselves or someone/something on their behalf. The monitoring requires real time information to be available at all point in time, which re-emphasizes the need for a good information system for a PLN.

In a PLN, a truck's route can change anytime and can be directed towards any DC if the truck is offered an appropriate amount. The higher amount offered by the packages at a DC, greater is the chance that the truck would choose a route towards that DC. This amount, in turn, depends on the money that the packages would be willing to spend for their journey. Since a package would also try to reduce its transportation cost (in terms of time or money, depending on the objective function of the package), the package would offer an amount depending on the configuration of the system. This amount depends on the following factors:

1. Value of the package (explained below)
2. How soon it wants to reach the destination
3. Number of trucks near the DC
4. Number of trucks coming towards the DC
5. Number of packages waiting at the DC
6. Money offered by other packages at the DC
7. Storage cost at the DC

The value of a package is the measure of the importance of the package at its destination. Higher the value of a package, the more important it is at the destination and hence the more money it offers to be transported so that it could reach its destination as soon as possible. Based on the above-mentioned factors, a package might offer less or more for the same journey. For example, a package might offer less money if there is an abundance of trucks. It might also offer less money of it is not in a hurry to reach its destination, but then the package would have to pay more money for the storage at the DC. It might offer large money if there are not many trucks available in the vicinity and the package has to pay higher amount to call the same truck, had if it been nearer to the DC. It might also offer higher amount if it is in a hurry to reach its destination because the more it waits at a DC, the more value it losses. So a package would re-optimize at every leg of its journey and the constraint under which it optimizes keeps on changing as the environment in which the package operates in keeps on changing. The money that a package offers to truck and DC are the *bids* it gives for services of the truck used for its transport and DC for the time it spend there and for any other services like loading and unloading. The approach of having each package bid for the services of the resources in the network was inspired by Mackie-Mason and Varian's [6] proposal to have each packet in the Internet bid for the use of routers when they are congested.

Kay and Jain [7] propose a potential solution to transport packages through a PLN by the use of software agents. This thesis focuses on exploring the implementation of agent-based co-ordination for package transportation through a PLN. The thesis concentrates on truck agents and their behavior to affect the performance of a PLN. The simplest form of package agent is assumed for the thesis. A package agent only calculates the bid amount for various legs of package's journey. It is assumed to be done at the beginning of the package journey and package agent does not make any other decision on behalf of its corresponding package. Chapter 2 focuses on the development of a simulation model of a PLN, where it is used

initially to determine an analytical formula to estimate the waiting time for packages at a DC. Chapter 3 and Chapter 4 of this thesis are focused on the protocols required to coordinate package transport through a multi-firm environment of a PLN. The simulation developed in Chapter 2 is used in these chapters to compare the performance of a PLN with and without the use of the protocols in order to estimate the potential benefits associated with their use.

2. Estimating Package Waiting Time

An analytical formula to calculate the average waiting time for packages, transported through PLN, was proposed by Kay and Parlikad [1]. The formula assumes that both trucks and packages arrive at random at a DC (i.e., a Poisson process, or exponential inter-arrival time) and that the expected waiting time for any package transported through a link, connecting two neighboring DCs, is the average headway (i.e., the average time between truck trips) on that link. For example, three trips per day on a link from DC*i* to DC*j* would imply an eight-hour average waiting time for every package at DC*i* transported through the link to DC*j*. But the assumption of a Poisson process for truck and package arrival at a DC is not realistic. After a new package is randomly generated at a DC, it departs in bulk with other packages that are to arrive in bulk at the next DC. This mix of bulk arrivals and departures makes it impossible to create a simple formula to estimate package waiting times [2].

This chapter is focused on improving the analytical formula used to calculate the average waiting time for the packages in the system by including the bulk arrival and departure of packages at DCs. Average waiting time is likely to be anywhere between the full headway (as assumed in [1]) and half the headway. The generation of package demand at any DC still follows Poisson process. The effect of bulk arrival of packages is studied using a simulation model of a PLN.

2.1 Simulation Model

A simulation model was prepared to study the effect of bulk arrival of packages on their waiting time in a PLN. Although the simulation could be used for any kind of network, this thesis focuses on the hypothetical network shown in Figure 1.2 and other smaller networks derived from it.

2.1.1 Overview

A PLN consists of several DCs; each fulfills the demands of its surrounding region. If demand for a package arises in a region, the DC in that region places an order at the same and other DC in the PLN. The DC which places the demand is called the *customer DC* and the DC where demand is placed is called the *supplier DC.* When the package corresponding to the demand is ready at the supplier DC, it is loaded onto a truck (if a truck is available at the DC; otherwise the package waits for a truck) and then it travels on the shortest-time route to the customer DC. The shortest route is determined using Dijkstra's algorithm, including the time required for unloading and loading at each intermediate DC.

The truck which carries the package takes it to the next, adjacent DC on its route to the customer DC. The package is then unloaded at the DC and the truck joins the queue of trucks at the DC. The package is later loaded on the same or different truck depending on what other trucks and packages are present at the DC. The truck queue is FIFO; thus the first truck in the queue gets the first chance to get packages. There are several queues of packages at a DC, one for each DC directly connected to the DC. A truck at a DC picks packages from the queue that has the maximum number of packages waiting and the packages in the picked queue are loaded on the truck based on their arrival time at the DC, earliest first. This process repeats until the package reaches the customer DC. Figure 2.1 describes the simulation and various stages of package transportation.

The simulation model of a PLN has been developed in MATLAB because it provides an easy way to store a large quantity of information in matrices and it has a very huge library of built-in functions that are extremely effective in matrix manipulations. The object-oriented concepts have also been used in the simulation to model some parts of the system. Because MATLAB is slower in accessing information from objects as compared

to getting information from matrices, only trucks, DCs and the event calendar to manage simulation has been modeled as objects since there are relatively fewer trucks and DCs in a PLN as compared to packages and only one object to manage events for simulation.

The model consists of the following classes:

1. *truck* – models truck's behavior in the system.
2. *supplierdc* – models the behavior of a supplier DC in the system.
3. *customerdc* – models the behavior of a customer DC in the system.
4. *plndc* – inherits the behaviors of *supplierdc* and *customerdc* in a PLN.
5. *eventcal* – manages the events in the simulation.

2.1.2 Input Parameters

Following are the main parameters used in the simulation:

1. **Transport Demand:** The demand at a DC is assumed to be proportional to the population of the region surrounding the DC. Kay and Parlikad [1] give the population of each region for the network of 36 DCs and the population as a percentage of the total population. The population of each region was estimated using the population of ZIP Code Tabulation Areas (ZCTA) surrounding the DC.
2. **Proximity Factor:** Transport demand to and from a pair of DCs is estimated by using the population percentage of each DC together with a proximity factor [1].
3. **Truck Capacity**
4. **Load Factor:** This is one of the most important factors in the design of the network. It tells, an average, how much of truck capacity is utilized. An 80% or 0.8 *load factor* means that system is designed in such a way that on average 80% of entire truck capacity is utilized.

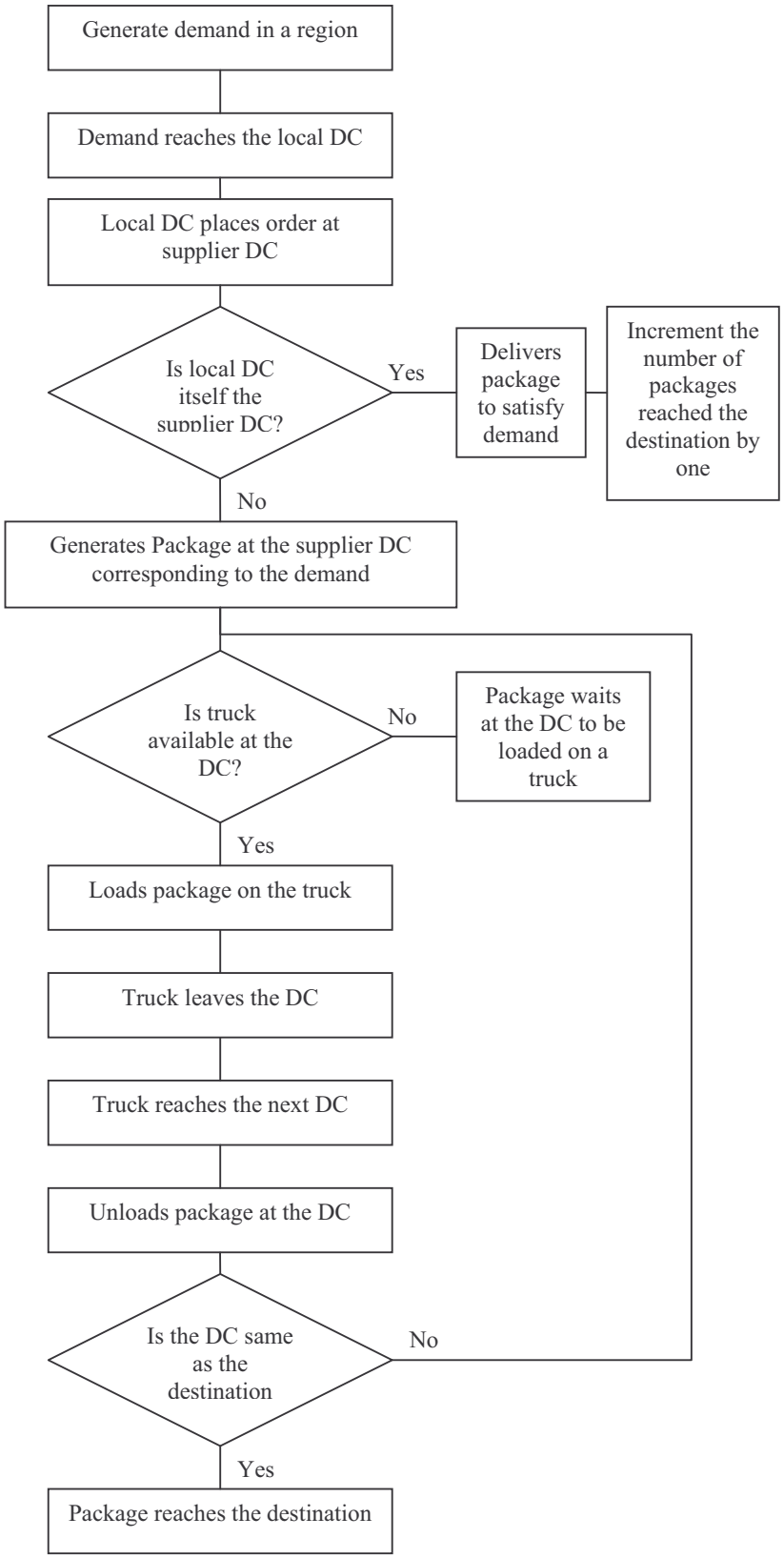


Figure2.1: Flowchart showing various stages of package transportation in the simulation model

5. **Total demand of the packages per day for the entire network**
6. **Truck Speed**
7. **Loading/Unloading Time for packages at a DC**
8. **Processing Time:** Time between arrival of demand at a DC and the corresponding package ready to be loaded on to a truck.

2.1.3 Assumptions

Following assumptions are used to make the simulation model of a Public Logistics Network:

1. *All trucks are of equal capacity*. PLN has been designed such that all trucks are of same capacity [1]. This also brings standardization to the system.
2. *Trucks move at a constant speed of 60 miles/hr between two DCs*: Between DCs, trucks would move on interstates where they could travel with an average speed of 60 miles/hr.
3. *The loading/unloading time at a DC is zero*: Kay and Parlikad [1] shows that loading/unloading time plays a major role in the performance of PLN as compared to other distribution systems. PLN works better with very low loading/unloading time.
4. *Each DC has infinite capacity.*
5. *Processing time is zero*: As soon as an order is placed at the supplier DC, a package becomes ready to be loaded on a truck immediately.
6. *Proximity factor is 0*: This makes each DC equally likely to get an order for a package from any DC, irrespective of distance between two DCs [1].

2.1.4 Random Number Generation

In the simulation, demand generation at a DC follows Poisson process. Since demand for each pair of DCs is different, different means are used to generate random numbers for each pair of DCs. The mean for a Poisson distribution for a pair of DCs was calculated using the

demand of packages for one day for that pair of DCs. If d_{ij} is the demand of packages per day at DC_i fulfilled by DC_j, the mean for Poisson distribution to generate demand is $\frac{d_{ij}}{24}$.

2.1.5 Execution of simulation

Once the variables, random number generation, event calendar and other objects are initialized, the actual execution of simulation starts. Apart from the events of end of warm up period and the end of simulation run, there are five main events in the simulation:

1. **Generation of demand:** *This event is identified by id 3* in the simulation. Each DC in a PLN sends demand for packages to every DC (including itself). When a demand is generated, it is assumed to reach the supplier DC instantaneously. An event is scheduled for the arrival of package at the supplier DC corresponding to the demand. Another event to generate demand at the customer DC is also scheduled for the combination of customer DC and supplier DC. Figure 2.2 shows the activities that take place at the occurrence of this event.

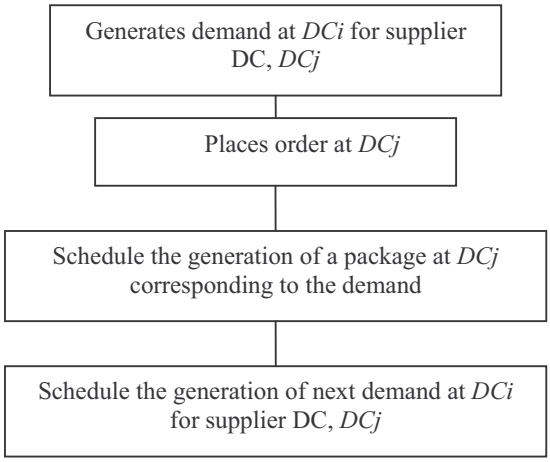


Figure2.2: Flowchart showing activities at the event of generation of demand

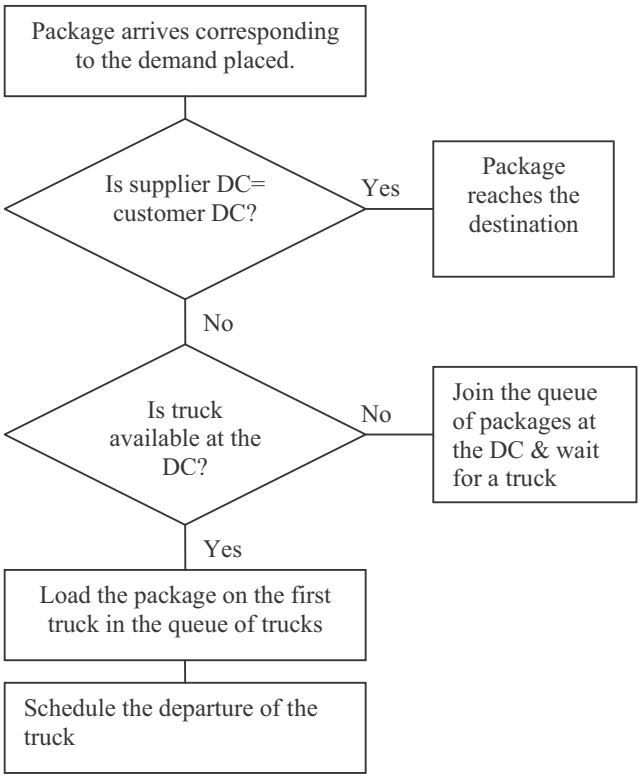


Figure2.3: Flowchart showing the activities at the event of package ready to be picked at the supplier DC

2. **Package ready to be picked at the supplier DC:** *This event is identified by id 4.* When a package is ready to be picked from the supplier DC, it can either find a truck waiting at the DC (in the truck queue) or the package has to wait for a truck. If the package finds the truck waiting, the departure of the truck is scheduled. Figure 2.3 shows the activities that take place at the occurrence of this event.
3. **Truck arrival at a DC:** *This event is identified by id 1.* When a truck arrives at a DC, it unloads the package it carries and then goes to the truck queue in the DC. If there are packages in the DC waiting to be picked up, the first truck from the truck queue is loaded with the packages. There can be many queues of packages in the DC, each corresponding to DC connected directly to the DC. The truck picks the queue with the maximum number

of packages. Packages from the queue are loaded in the order of

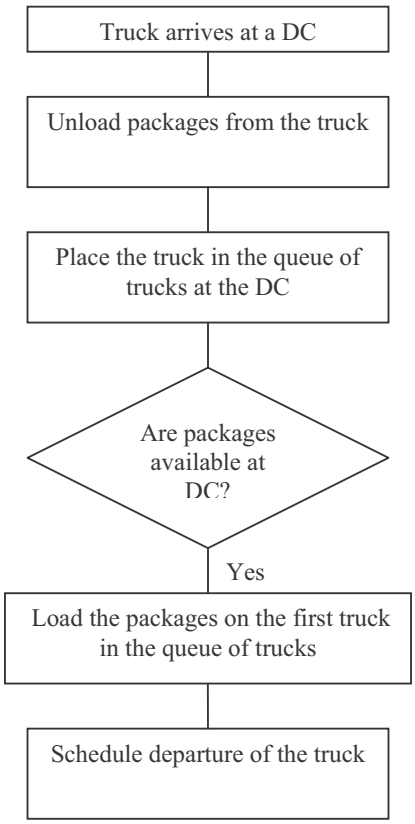


Figure2.4: Flowchart showing the activities at the event of truck arrival at a DC

their arrival at the DC, earliest first. The number of packages loaded on the truck is the minimum of the truck capacity and the number of packages in the selected queue. After the loading is done, the departure of the truck is scheduled. Figure 2.4 shows the activities that take place at the occurrence of this event.

4. **Truck leaving a DC:** *This event is identified by id 0.* At this event, truck leaves a DC and the event of truck arrival at the next DC is scheduled by calculating the travel time and adding the current simulation time to it. Figure 2.5 shows the activities that take place at the occurrence of this event.

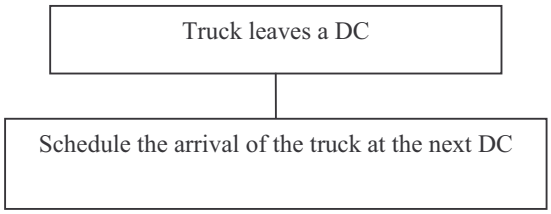


Figure2.5: Flowchart showing activities at the event of truck departure

5. **Drawing of graphs:** It is identified by id 5.

2.2 Analytical Formula

The simulation described in the previous section was used to simulate the network shown in Figure 1. All the 36 DCs were used in the simulation. Following are the steps used to determine an analytical equation to calculate waiting time using various simulation runs:

Step 1:

The simulation was run for the set of values shown in Table 2.1. A load factor between 0.5 and 0.85 was considered because, over a long run, it is not economic for PLN to operate at very low load factor and a very high load factor results in fewer trucks and hence larger queues of packages waiting at DCs.

For various simulation runs, net flow through various links (connecting two DCs) called link flow and the average waiting time for packages along these links were recorded. Link flow for a link is defined as the ratio of number of packages transported through the link to the total number of packages transported through different links in the network. Also, the overall average waiting time and overall average transportation time were recorded for different combinations shown in Table 2.1.

Table2.1: Set of values for simulation runs

Demand of Packages per day	Truck Capacity	Load Factor	Number of Replications
30,000	20	0.7	1
40,000	20	0.7	1
50,000	20	0.5	3
		0.55	1
		0.6	1
		0.65	1
		0.7	11
		0.72	1
		0.75	1
		0.77	1
		0.8	2
		0.82	1
		0.85	1
70,000	20	0.5	1
		0.7	1
		0.8	1
100,000	20	0.7	1
50,000	100	0.5	1
		0.7	4
		0.8	1
70,000	100	0.7	6
		0.8	6
150,000	100	0.7	9
		0.8	7

Table 2.2 and 2.3 show some of the results obtained from the simulation:

Table2.2: Average Waiting Time and Average Transportation Time (in hours) for demand of 50000packages per day, truck capacity 20 and load factor 0.7

Seed for Random Number Generation	Average Waiting Time (overall)	Average Transportation Time (Overall)
300	0.2424	6.931
6507440	0.2425	6.936
8661526	0.2423	6.9381
7976494	0.2424	6.9336
1568674	0.2426	6.9363
415885	0.2424	6.9381
4121782	0.2427	6.9355

Table2.3: Average Waiting Time and Average Transportation Time (in hours) for demand of 70000packages per day, truck capacity 100 and load factor 0.8

Seed for Random Number Generation	Average Waiting Time (overall)	Average Transportation Time (Overall)
6721375	1.1276	7.8021
8317961	1.1398	7.4433
4610952	1.1325	7.5669
502689	1.1313	7.6591
7679504	1.1019	8.0812
4983114	1.1411	7.2729

Step 2:

After plotting average waiting time (for each link) versus link flow for all the combinations mentioned in Table 2.1, it can be noticed that for very small arc demand, the waiting time is very large and as the arc demand increases, the waiting time decreases.

Figure 2.6 shows one such plot of link average waiting time versus link flow. Looking at the R-square value, we can conclude that it is good fit. It was noticed that for all the combinations mentioned in Table 2.1, the graph is similar to the one shown in Figure 2.6.

So the equation to calculate link average waiting time could be written in the form

$$w = c \cdot x^{-a} \qquad (1)$$

where x - link flow

w - link average waiting time

c and a are parameters

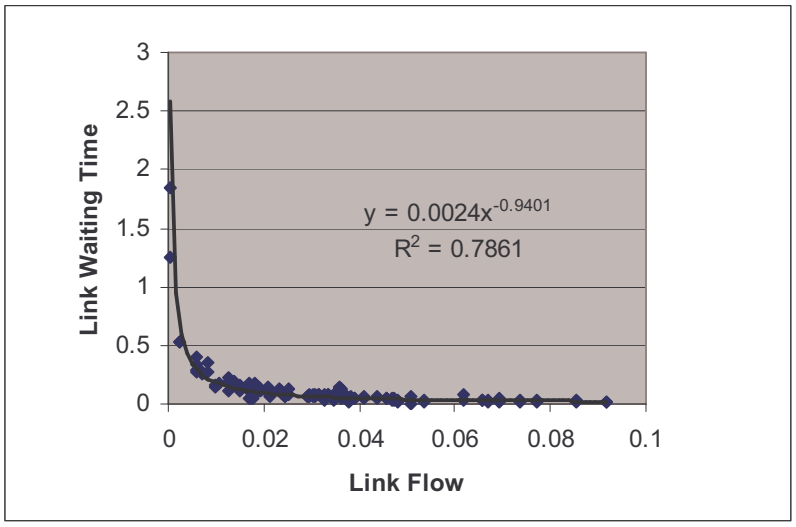


Figure2.6: Plot of Average Link Waiting Time Vs Net Link Demand for demand of 50000 packages per day, truck capacity 20 and load factor 0.7

Step 3:

On further analysis of the simulation results, it was found that parameter *c* in Equation (1) is of the form:

$$c = \left(\frac{tc}{np}\right) * d \qquad (2)$$

where *tc* - truck capacity
np - Number of packages per day
d - A parameter.

Table 2.4 shows the relation given by equation (2). From at Table 2.4, it can be noticed that *d* value is nearly same for same load factor and does not have the effect of number of packages per day and the truck capacity. This helps to conclude that *d* in Equation (2) is a function of only load factor.

Table2.4: Shows relation between parameters 'c' and 'd' for different sets of demand, truck capacity and load factors

Demand of Packages per Day	Truck Capacity	Load Factor	Coefficient (c) (from simulation)	Truck Cap/Demand of Packages (tc/np) = r	d = c/r
30000	20	0.7	0.0038	0.000666667	5.7
40000	20	0.7	0.0029	0.0005	5.8
50000	20	0.5	0.0015	0.0004	3.75
50000	20	0.5	0.0015	0.0004	3.75
50000	20	0.55	0.0016	0.0004	4
50000	20	0.6	0.0018	0.0004	4.5
50000	20	0.65	0.0021	0.0004	5.25
50000	20	0.67	0.0022	0.0004	5.5
50000	20	0.7	0.0024	0.0004	6
50000	20	0.7	0.0023	0.0004	5.75
50000	20	0.7	0.0024	0.0004	6
50000	20	0.7	0.0024	0.0004	6
50000	20	0.7	0.0023	0.0004	5.75
50000	20	0.7	0.0024	0.0004	6
50000	20	0.7	0.0024	0.0004	6
50000	20	0.7	0.0024	0.0004	6
50000	20	0.72	0.0025	0.0004	6.25
50000	20	0.75	0.0027	0.0004	6.75
50000	20	0.77	0.0029	0.0004	7.25
50000	20	0.8	0.0032	0.0004	8
50000	20	0.8	0.0032	0.0004	8
50000	20	0.82	0.0034	0.0004	8.5
50000	20	0.85	0.0038	0.0004	9.5
70000	20	0.5	0.0011	0.000285714	3.85
70000	20	0.7	0.0017	0.000285714	5.95
70000	20	0.8	0.0023	0.000285714	8.05

Step 4:

To find the relation between *d* and the load factor *lf*, the values of *d* was plotted for different values of load factor for different set of truck capacity and number of packages per day. Different types of equations were tested on the plot.

Table2.5: R-square values for different equations on figure 2.7

Equation	R-square
Linear	0.9262
Quadratic	0.9956
Cubic	0.9961
Exponential	0.983

Table 2.5 shows the R-square for four equations having relation between d and load factor. It can be noted from Table 2.5, that the quadratic, cubic and exponential equations have higher R-square value as compared to the linear equation. Since R-square value for last three equations are closer to 1 than the linear equation, one of the last three equations would be a good fit. But exponential equation was chosen over quadratic and cubic equation because the exponential equation has less parameter as compared to the other two. Figure 2.7 shows the plot of *d* versus load factor and the exponential equation on it.

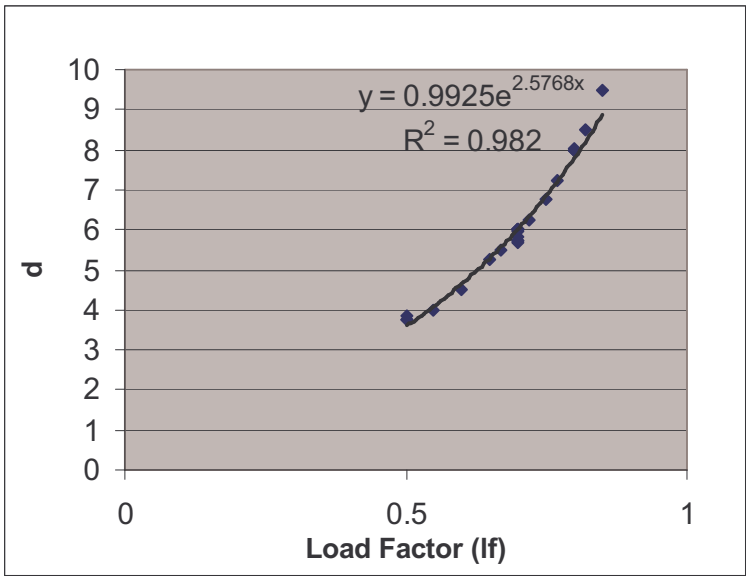


Figure2.7: Exponential Equation fitted on the plot between 'd' and load factor

In Figure 2.7, *y* stands for *d and x* stands for *lf*. So the relation between *d* and *lf* can be written approximately as:

$$d = e^{2.6lf} \tag{3}$$

Combining Eqs. (1), (2) and (3), equation for link average waiting time can be written as:

$$w = \left(\frac{tc}{np}\right) \cdot e^{2.6lf} \cdot x^{-a} \tag{4}$$

Step 5:

Comparing arc waiting time obtained from simulation with the formula given by Equation (4), for various values of *a*, the values fit best for $a = 0.96$.

So the final equation to calculate link average waiting time can be written as:

$$w = \left(\frac{tc}{np} \right) \cdot e^{2.6lf} \cdot x^{-0.96} \qquad (5)$$

The average waiting time (overall) can then be written as:

$$AW = \Sigma \; w_i x_i \qquad (6)$$

where AW - Average Waiting Time (overall)

w_i - Average Waiting time for link 'i'

x_i - Net flow for link 'i'.

2.2.1 Verification

The mathematical formula was verified with the help of simulation. Two approaches were used to verify the model:

- By reassigning the demand among DCs randomly.
- By reassigning the distances between DCs randomly and also reassigning the demand among DCs randomly.

Approach 1: Reassigning the demand among DCs randomly

Table2.6: Verification of the Equation (4) by assigning demand randomly among DCs

Demand of packages per day	Truck Capacity	Load factor	Avg. Waiting Time (from Simulation)	Avg. Waiting Time (from new analytical model)
50,000	20	0.5	0.1609	0.1493
50,000	20	0.7	0.2479	0.2516
50,000	20	0.8	0.3195	0.3257
50,000	100	0.7	1.2565	1.2570
50,000	100	0.8	1.6032	1.6274
70,000	100	0.7	0.8990	0.8976
70,000	100	0.8	1.1290	1.1636
150,000	100	0.7	0.4178	0.4188
150,000	100	0.8	0.5289	0.5443

Approach 2: Reassigning the distances between DCs randomly and also reassigning the demand among DCs randomly.

Table2.7: Verification of the Equation (4) by reassigning distances between DCs and also demand among DCs randomly

Demand of packages per day	Truck Capacity	Load factor	Avg. Waiting Time (from Simulation)	Avg. Waiting Time (from new analytical model)
50,000	100	0.8	1.5499	1.5649
70,000	100	0.7	1.0823	1.0824

2.3 Waiting Time from simulation versus Headway

Table 2.8 shows, for different sets of "demand of packages per day", "truck capacity" and "load factor", values of waiting time from simulation, waiting time from analytical formula [1] with half headway and waiting time from the formula [1] with full headway (headway is described at the beginning of this chapter and half headway is 0.5 times full headway). It can be noticed from the Table 2.8 that for the set of values considered, waiting times from simulation are closer to waiting time from analytical model with half headway than waiting time from analytical model with full headway.

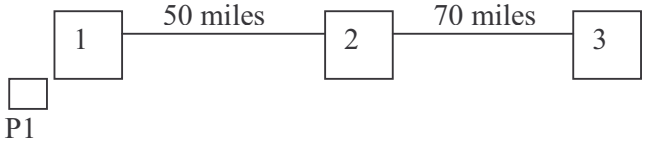
Table2.8: Comparison of "Waiting Time from simulation" with Half Headway and Full Headway

Demand of Packages per day	**Truck Capacity**	**Load Factor**	**Waiting Time (from Simulation)**	**Waiting Time (Half Headway)**	**Waiting Time (Full Headway)**
100,000	20	0.7	0.1213	0.1982	0.3965
30,000	20	0.7	0.4046	0.6608	1.3216
40,000	20	0.7	0.3028	0.4956	0.9912
50,000	20	0.5	0.1551	0.2832	0.5664
50,000	20	0.7	0.2424	0.3965	0.7930
50,000	20	0.8	0.3109	0.4531	0.9062
70,000	20	0.7	0.1735	0.2832	0.5664

3. Protocol Design

A Public Logistics Network has been envisioned as decentralized in nature with multiple firms owning different parts of the system. For example, a company could own one or more trucks, another company could own one or more DCs, and yet another company could own a combination of trucks and DCs. The presence of multiple companies would make the coordination of package transport more difficult than any private logistics network, where a single company provides the centralized control of the entire network. With an aim to achieve a certain level of coordination among different entities of a PLN so that packages that value transport the greatest are able to be matched with trucks that can provide the transport at the least cost, a set of protocols has been proposed by Kay [4].

The protocols assume that both trucks and packages have intelligence in the form of software agents and both truck and package agents make decisions to affect the performance of a PLN. However, this thesis focuses on truck agents and how their behavior affects the system. As a result, only those protocols that are required for truck agents to make decisions are considered in this chapter. The package agent is used only to decide its bid amount for its corresponding package for each leg of its journey. A bid amount is the money offered by a package agent to transport the corresponding package to the adjacent DC. Figure 3.1 shows how a package agent decides its bid amount for a package transported between DC1 and DC2, and DC2 and DC3. The bid amount for each package is calculated at the start of its journey and agent makes no other decisions on behalf of the package throughout its journey. The package agent also provides information to a DC about the package's bid amount and its arrival time at the DC. Since this thesis considers the simplest behavior of package agent, withdrawal and re-bidding by package agents [4] is not considered here.



Money for transportation for P1 = $40

Bid amount for P1 from DC1 to DC2 ⟶ (50)/ (50+70)*40= $16.67

Bid amount for P1 from DC2 to DC3 ⟶ (70)/ (50+70)*40= $23.33

Figure3.1: Bid amount for package P1 from DC1 to DC2 and from DC2 to DC3

3.1 Basic Definitions

In order to explain the terms, it is assumed that all trucks are identical with a capacity of three.

1. *Reservation*: Truck is said to have made *reservation* for a load of packages at a DC, when it agrees to carry those packages from that DC to the next one. A reservation is valid until the truck reneges the packages or the packages are loaded on to that truck.
2. *Load*: A load, for a queue of packages, is defined as the number of packages in each queue less than or equal to the truck capacity, which ever is more. If number of packages in a queue is more than truck capacity, load will have packages equal to truck capacity. Among all the packages in a queue, the packages in a load get first chance to be accepted by a truck. If a truck accepts a load, some or all of the remaining packages (depending on number of packages in the queue) form a new load. When a truck reneges packages belonging to a queue, the load formation for that queue is re-evaluated to examine the possibility of reneged packages being part of the load. Figure 3.2 shows load formation in various queues.

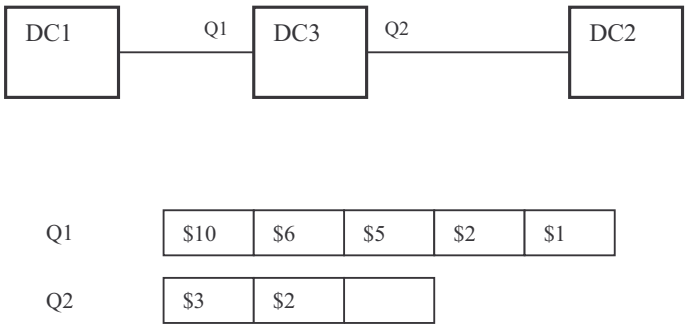


Figure3.2: Load formation at DC3

In Figure 3.2, Q1 is the queue of packages waiting at DC3 to go to DC1, Q2 is the queue of packages waiting at DC3 to go to DC2 and packages in each queue are arranged in the decreasing order of their bids. Since the capacity of a truck is 3, maximum number of packages in any load is 3. If load is formed based on values, the load for Q1 will have packages with values $10, $6 and $5 and the load for Q2 will have packages with values $3 and $2. Load also consists of incoming packages and load formation at a DC takes time into account (see Figure 3.3).

3. *Accepted Load*: Load at a DC that a truck agrees to carry to the next DC is the *accepted load* at that DC for that truck. For example, in Figure 3.2, if a truck accepts to carry the load for Q1, that load becomes the accepted load at DC3 for that truck.
4. *Load bid*: The amount offered by a load is referred to as *load bid*. In Figure 3.2, load bid for Q1 is $21 ($10 + $6 + $5) and the load bid for Q2 is $5 ($3 + $2).
5. *Accepted Load bid*: The amount offered by the accepted load to the truck is the *accepted load bid*.

3.2 Package Protocol

The package protocol determines which packages are selected to join a load. The goal of package selection is to encourage a package to submit a bid that represents its true value for transport as soon as possible, thereby allowing trucks to be more responsive.

3.2.1 Load Formation

Packages assigned to load that maximizes resulting load bid. Packages at or inbound to a DC can participate in bidding process at the DC.

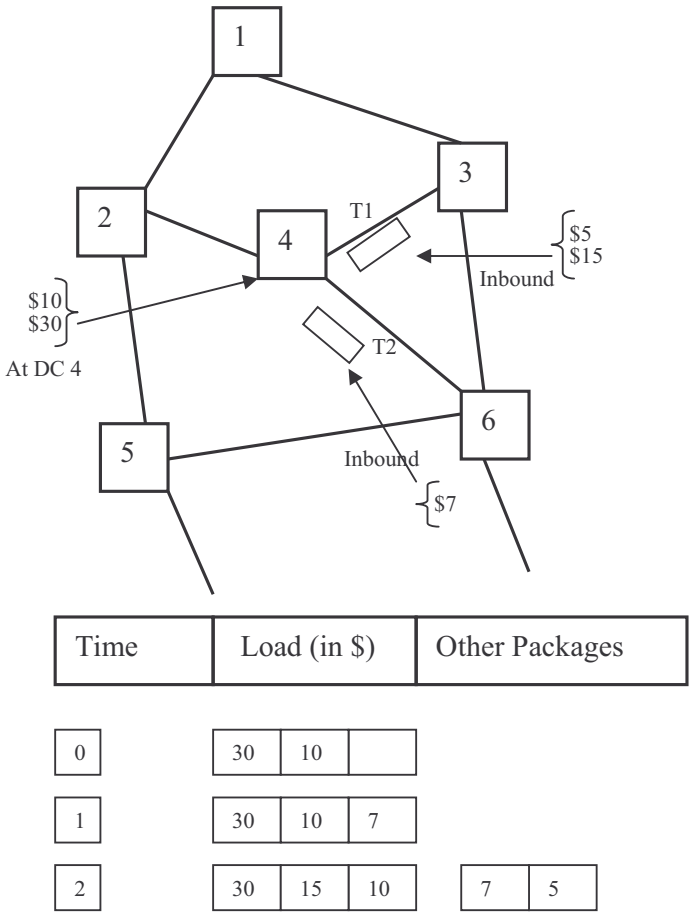


Figure3.3: Load formation at DC4 at time t = 0, 1 and 2. Trucks T1 and T2 will reach DC4 at time t = 1 and t = 2 respectively

Inbound packages are considered future inventory, the arrival time of which is known, so they can also influence a truck's behavior and hence part of the bidding process. Since all trucks are assumed to be same, maximum load size is same as the capacity of any truck. Figure 3.3 shows load formation at various times at DC4, assuming that the maximum load size is 3.

3.2.2 Allocation of Load Bid

Truck's portion of a load bid does not increase after acceptance: The portion of a load bid given to the truck that accepted the load is equal to the load bid at the time of acceptance; all subsequent increases in the load bid are given to the packages that were in the load at the time of acceptance (and remain in the load) in proportion to their bid amounts.

By not increasing the amount of truck's portion of a load bid, a truck has no incentive to wait at any DC for more packages to join the load. This results in more predictable arrival and departure times. By giving the increase in load bid to the packages who were present at the time of acceptance of the load (and remain in the load), it encourages early bidding by the packages and hence influence a truck's behavior.

The load bid can increase for one of the following two reasons:

1. Addition of a package because load size is less than the capacity of a truck.
2. A new package of high value replacing a package of low value in the load, when the load size is same as the capacity of a truck.

Table 3.1 shows increases in load bids, cost to different packages and truck's share and response to the increase in bid. Assuming the truck capacity of 3 packages and it requires minimum of $10 to accept bid, the truck first rejects the load because its bid is $7. When the second package arrives, the load bid becomes $10 ($7 + $3) and the truck accepts the

load. The third package of value $5 arrives and because of space available for one more package in the load, it becomes a part of the load. The increase in load bid i.e. $5 ($15 – $10) is divided between packages that were in the load at the time of load acceptance, the $7- and $3-bid packages, based on their 70% and 30% portions of the original load bid, so their costs are now $4 and $1, respectively. When the $6-bid package is added to the load replacing $3-bid, all of the additional $8 is allocated to the $7, making its journey free and it also makes $1.

Table3.1: Allocation of load bid.

Package Event	Truck Response	Load Bid	Truck Portion	Allocated Portion	Load (Bid/Cost)
Bid & Join	Reject	7	7	0	7/7 | |
Bid & Join	Accept	10	10	0	7/7 | 3/3 |
Bid & Join	–	15	10	5	7/4 | 5/5 | 3/1
Bid, Join & Drop	–	18	10	8	7/-1 | 6/6 | 5/5

3.3 Truck Protocol

The truck protocol, followed by every truck of the system, is used to determine which is used to transport what loads at what DC.

3.3.1 Priority for Accepting Loads

Opportunity to accept or reject load based on truck's arrival time at DC: Priority to accept or reject an available load at a DC is first given to trucks already at the DC based on their order of arrival, earliest first, and then to trucks not at the DC based on their expected time of arrival at the DC along their intended path to the DC (see Figure 3.4). If

all the trucks, which are given chance, reject a load, it is posted at the DC and is then available for any truck to accept.

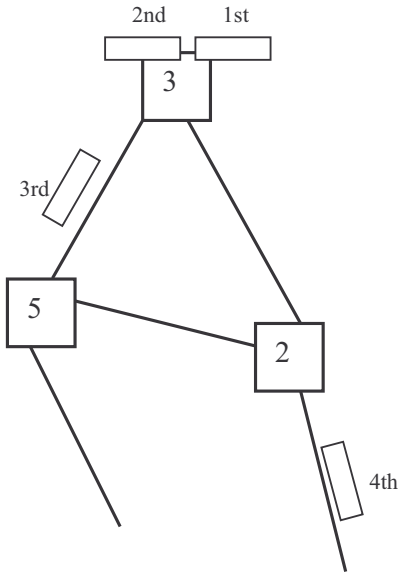


Figure3.4: Priority to accept and reject load at DC3

3.3.2 Loads at DCs along the Intended path

A load must be accepted at each DC along truck's intended path. Before accepting a load at a DC, a truck needs to accept load at each intermediate DC along its intended path to the DC (see Figure 3.5).

This requirement can result in a truck accepting low bids at intermediate DCs if the load bid at the last DC is very high, making it possible for some packages to pay very low money for transportation. This is especially good for the low value packages waiting at a DC because load bid at that DC is not sufficient enough for any truck to reserve the packages. But because of arrival of a high value package at some other DC, these low value packages could get chance to be on a truck (see Figure 3.6).

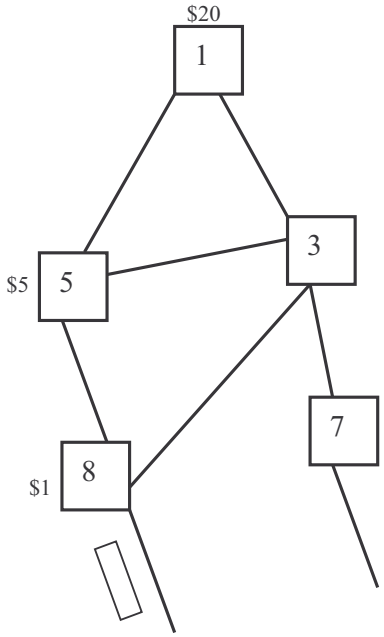


Figure3.5: Before accepting load at DC1, truck needs to accept load at DC5 and DC8

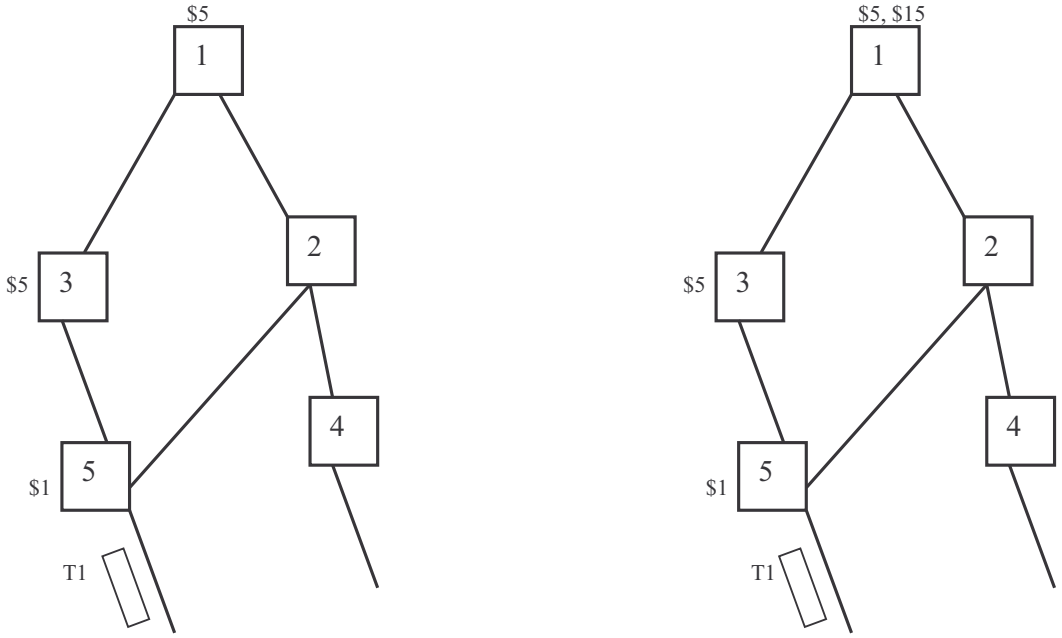


Figure3.6: Initially, load bids at DC5, DC3 and DC1 are low enough for truck T1; assuming truck requires $7 to travel between any two DCs. After $15, truck T1 accepts the bids at DC1 because it provides money for other links.

3.3.3 Intended Path

Shortest route to a DC. The intended path of the truck to a DC is the shortest route in terms of time from its current location to the DC. This results in trucks reaching DCs with reserved packages as soon as possible. This also helps in prioritizing trucks for *Truck Protocol Rule 1*, above.

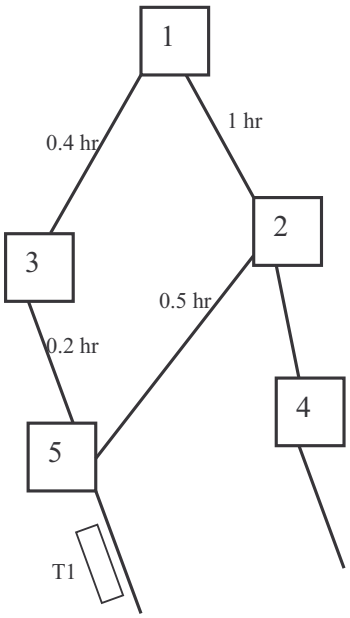


Figure3.7: Different routes possible for T1 to reach DC1

Figure 3.7 shows two possible routes for truck T1 to reach DC1. The first route takes 0.6 hours (DC5 to DC3 takes 0.2 hours and DC3 to DC1 takes 0.4 hours (loading/unloading time at each DC is assumed to be 0)). The second route takes 1.5 hours (DC5 to DC2 takes 0.5 hours and DC2 to DC1 takes 1 hour). Since first route is shorter in terms of time, truck T1 would be on it to reach DC1.

Since a truck needs to be at the DC in the shortest possible time, this could results in some inefficiency. Following scenario illustrates this.

Scenario 1:

Assumption: Transportation cost = $2/mile

Truck Speed = 60 miles/hr

Consider a truck T1 starting to move from DC6 to DC1 (see Figure 3.8). It would reach DC1 after 2 hrs. It has a package P1 with value $120, whose final destination is DC3. There is another truck T2 standing at DC5 because it is not able to find sufficient money to move. Since T2 is closer to DC1 than T1, T2 would get first chance to accept or reject load bid (*Truck Protocol Rule 1*).

From Figure 3.8, there could be two possible routes for truck T2 to reach DC1 because T2 would reach DC1 before P1 arrives by following any of the two routes. The possible routes and time taken by truck T2 to reach DC1 are shown in Table 3.2.

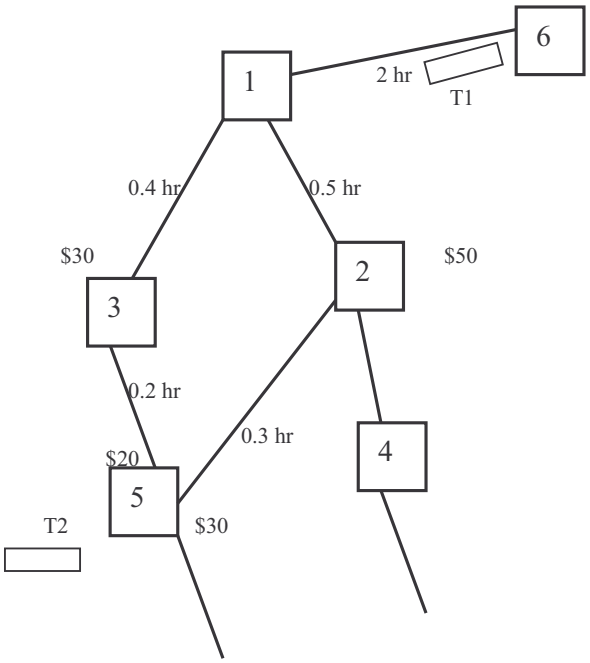


Figure3.8: Trucks end up getting less money because of shortest route constraint

Table3.2: Different possible routes for truck T2 to reach DC1 from DC5 and time taken to reach DC1

Route No.	Route	Time Taken
1	DC5 – DC2 – DC1	0.8 hrs (0.3 hrs on DC5 to DC2 + 0.5 hrs on DC2 to DC1)
2	DC5 – DC3 – DC1	0.6 hrs (0.2 hrs on DC5 to DC3 + 0.4 hrs on DC3 to DC1)

According to *Truck Protocol 2*, T2 should accept loads at intermediate DCs before accepting loads at DC1. Table 3.3 shows the money truck would get from different routes to DC1.

Table3.3: Different possible routes for truck T2 to reach DC1 from DC5 and money it would receive from them

Route No.	Route	Time Taken
1	DC5 – DC2 – DC1	$80 ($30 on DC5 to DC2 + $50 on DC2 to DC1)
2	DC5 – DC3 – DC1	$30 ($10 on DC5 to DC3 + $20 on DC3 to DC1)

So if truck T2 reserves package P1 (whose final destination is DC3) at DC1, T2 would travel till DC3. Table 3.4 shows complete routes, total money offered by packages on each route, transportation cost associated with each route and the net amount received by truck T2.

Table3.4: Complete route, money offered by packages on each route, transportation cost and net amount received by truck T2

Route No.	Route	Money offered by packages(A)	Transportation Cost (B)	Money received by truck T2 (A - B)
1	5–2–1–3	$200 ($30 + $50 + $120)	(0.3 hr + 0.5hr + 0.4 hr) × 60 miles/hr × $2/mile = $144	$56
2	5-3-1-3	$150 ($20 + $30 + $120)	(0.2 hr + 0.4 hr + 0.4 hr) × 60 miles/hr × $2/mile = $120	$30

Although truck T2 would get more money by going through route 1, it would be forced to go through route 2 because it is shorter (*Truck Protocol* 3).

3.3.4 Reneging

A truck can renege on its acceptance of a load at anytime. All the loads beyond the reneged load along the truck's intended path are also reneged. This provides flexibility to the trucks to move towards a place where they could make more profit. It also causes uncertainty in the route of a truck and hence its arrival time at a DC.

3.4 Framework for Agent Co-ordination

Each package and each truck is controlled by a software agent. The agents interact with each DC via the service provided by the DC (see Figure 3.9). As soon as a package leaves a DC, the package agent provides following information to the intermediate DC to which package is moving:

1. Arrival time of the package at the intermediate DC
2. Bid amount for the next leg of the journey starting from the intermediate DC
3. Next DC ID from the intermediate DC towards the destination.

The DC maintains the package information and uses it to form package loads and their corresponding bids. The DC also maintains trucks locations and tracks all of the truck paths in order to determine the expected arrival times of the trucks at the DC. Whenever there is a change in load bid, the DC determines the order in which trucks are offered the opportunity to accept or reject the load. If all the trucks reject the load, the load bid is posted at the DC with other active load bids. If any truck accepts the load, the DC maintains the association of the load and the truck. It also updates the active load bids and the association between loads and trucks whenever any truck reneges.

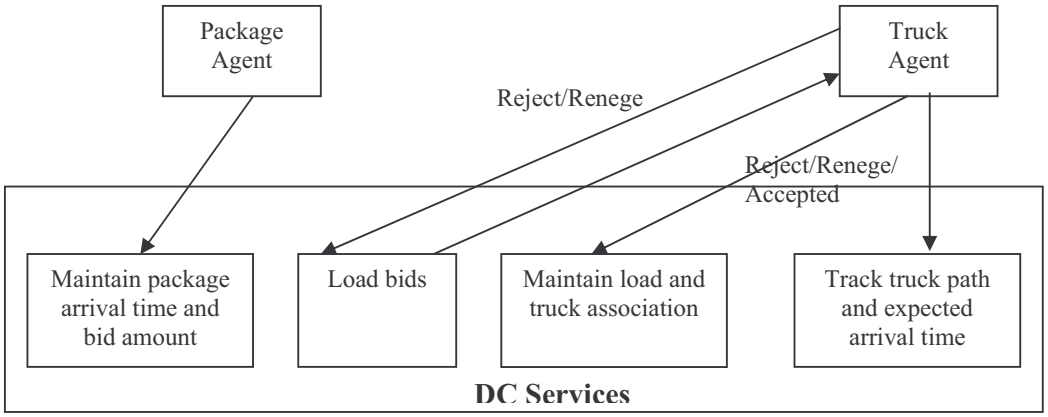


Figure3.9: Framework for the agent based co-ordination

3.5 Simulation

The simulation model described in Chapter 2 has been modified to implement the protocols mentioned in this chapter. Every time a load bid changes, the DC sends the new load bid to trucks in the order of their expected arrival time at the DC until it is accepted by a truck. Load bids can change for any of the following reasons:

1. A package appears at the supplier DC to satisfy demand at the customer DC.
2. A truck carrying packages starts moving towards a DC. Since load formation also includes incoming packages (Figure 3.3), packages starting from a DC towards an adjacent DC can change the load formation and hence the load bid at the adjacent DC.
3. A truck reneges a load at a DC.

In the simulation model, changes have been made in the following events to implement protocols:

1. **Package ready to be picked from the supplier DC:** *This event is identified by id 4.* When the package is ready to be picked from the supplier DC, the DC checks if the new package changes the active load and hence the load bid. If it does, DC sends

the new load bid to trucks in the order of their expected arrival time, until it is accepted by a truck. If no truck accepts the load, the load is posted with other active loads. If a truck accepts the load, this might cause the truck to renege the load at same or some other DC. This could cause change in the loads at those DCs and those DCs in turn sends the new load bids to the trucks, again in the order of their expected arrival time at those DCs. This goes on until there are no more changes in active loads at any DC in the system. During the entire process, if any truck is standing at one of the DCs where it has reservation, it is loaded with the package and the event of departure of the truck is scheduled at that DC. Figures 3.10 and 3.11 describe the activities that take place at this event.

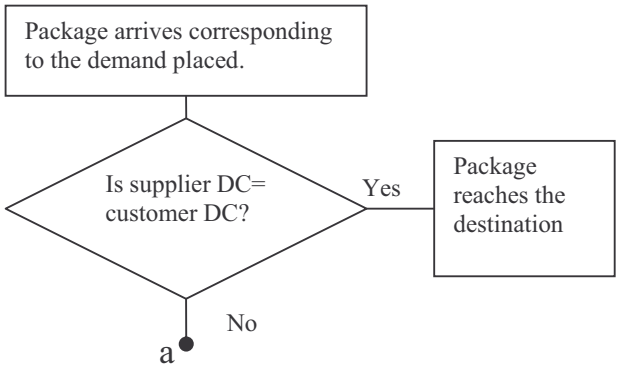


Figure3.10: Package arrival event in the simulation with protocol

In Figure 3.11, *m* stands for number of DCs where loads change either because of truck reneging or accepting loads at those DCs. *j* keeps track of the number DCs that have sent new load bid information in the algorithm. *i* keeps track of the number of trucks to which new load bid has been sent in the algorithm.

2. **Truck leaving the DC:** *This event is identified by id 0.* At this event, a truck leaves the DC and the event of truck arrival at the next DC is scheduled by calculating the

travel time and adding the current simulation time to it. The information about the package is sent to the next DC and, if it causes changes in the load formation at time equal to or greater than the truck arrival time, the whole process mentioned under "Package ready to be picked from the supplier DC" starts. Figure 3.12 and 3.11 shows the activities at this event.

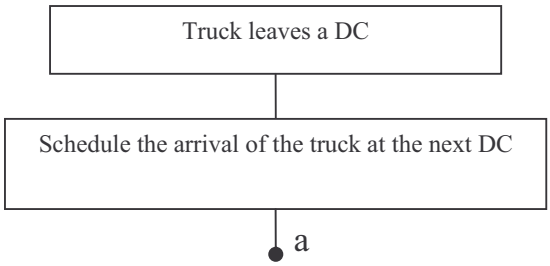


Figure3.11: Event of truck leaving a DC in the simulation with protocol

3. **Truck arriving at a DC:** When a truck arrives at a DC, if truck has reservation with the DC, the packages are loaded and an event to depart the truck is scheduled. Figure 3.13 shows the activities at this event.

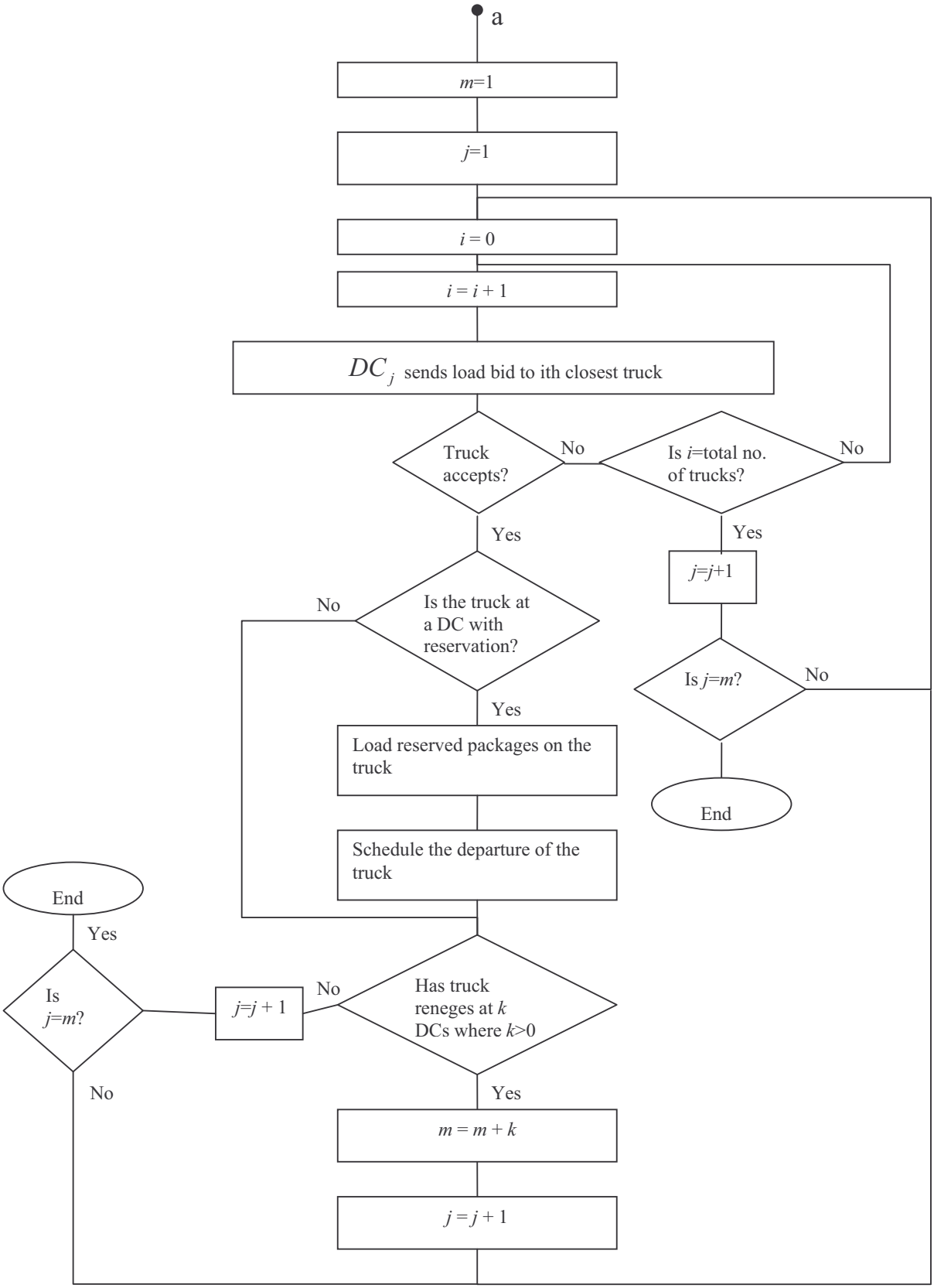


Figure3.12: Bidding Process

3.5.1 Assumptions

Following assumptions are used to make the simulation model of a Public Logistics Network implementing protocols:

1. *All trucks are of equal capacity*: Explained in Chapter 2
2. *Trucks move at an constant speed of 60miles/hr between two DCs*: Explained in Chapter 2
3. *The loading/unloading time at a DC is zero*: Explained in Chapter 2
4. *Storage cost at a DC is very high*: Explained in Chapter 2
5. *Each DC has infinite capacity*: Explained in Chapter 2
6. *Processing time is zero*: Explained in Chapter 2
7. *Proximity factor is 0*: Explained in Chapter 2
8. *Labor cost is very high*: This causes trucks to leave a DC to the adjacent DC as soon as they get a minimum amount from the packages that, e.g., could cover the fuel cost for that leg of journey.

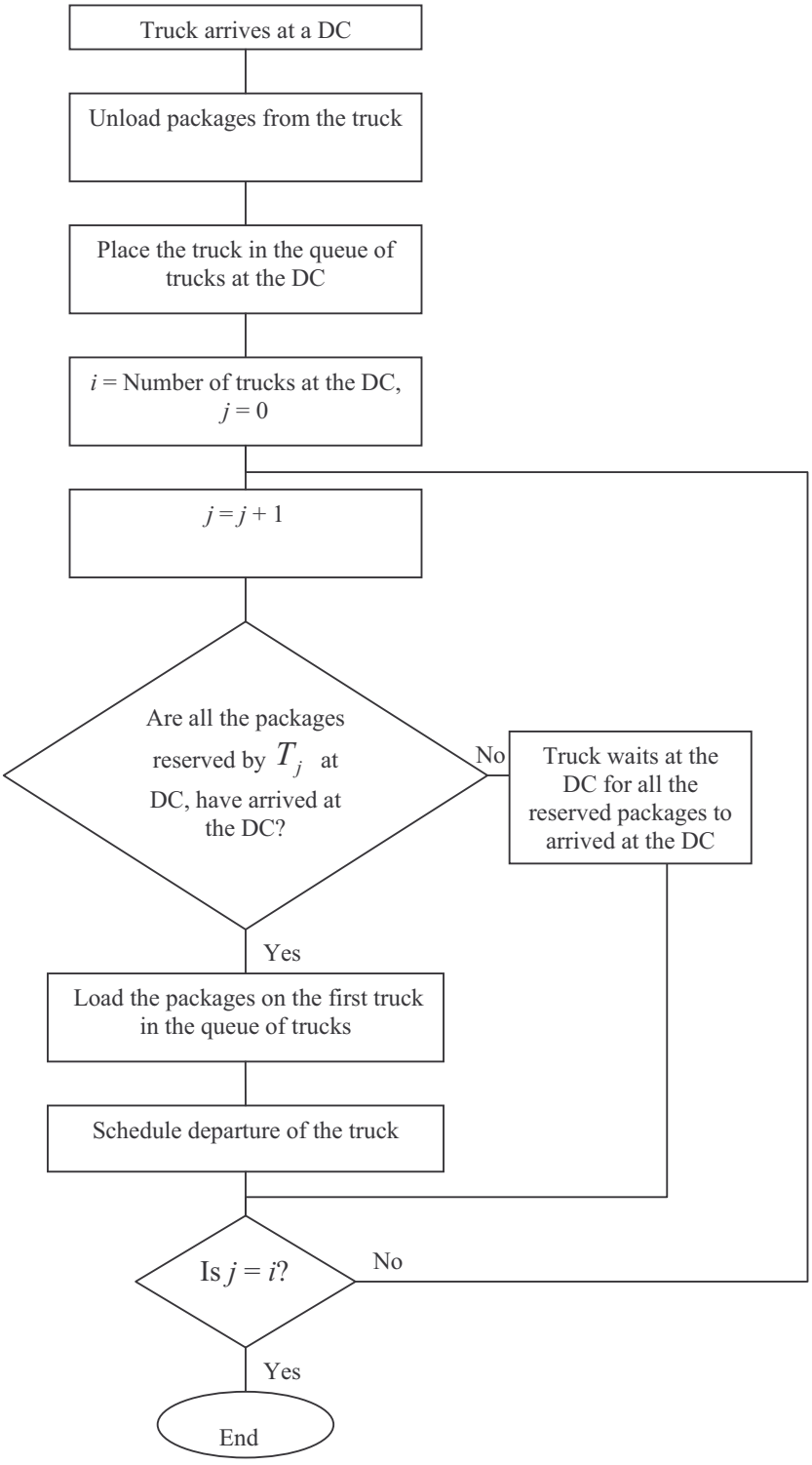


Figure3.13: Event of truck arriving at a DC in the simulation with protocol

4. Results and Analysis

4.1 Overview

A simulation of a PLN consisting of 36 DCs (see Figure 1.2) was run. To see the effectiveness of protocol in package transport, the results of the simulation with the protocol were compared with the results of a simulation of the PLN without the protocol.

1. **Without Protocol**: As the name suggests, this condition does not include any protocol, followed by any entity in the system. Trucks make decisions only when they are at DCs. At a DC, a truck picks the queue with maximum number of packages and packages are loaded on the truck in order of their arrival at the DC, earliest first. Only when a truck gets more money than the transportation cost, will it move from the DC. Figure 4.1 describes how a queue of packages is picked and packages are chosen to be loaded on a truck.
2. **With Protocol**: All the protocols mentioned in Chapter 3 were implemented in a PLN.

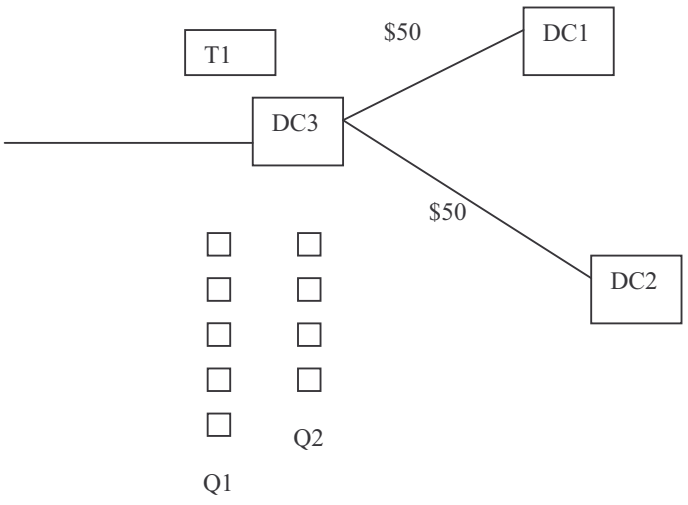


Figure4.1: Packages at DC3 waiting to be transported to their respective next DCs

Figure 4.1 shows two queues, Q1 and Q2, of packages waiting to be transported to DC1 and DC2, respectively, and truck T1 (assumed to have capacity 3) is at DC3. Any truck moving towards either DC1 or DC2 requires at least $50. Table 4.1 and Table 4.2 show packages in Q1 and Q2 at DC3 with their time of arrival (in the increasing order) and amount that they offer to a truck to move to the next DC. Assume that the current simulation time is 3 hrs.

Table4.1: Packages in Q1 at DC3 and their time of arrival (in the increasing order) and amount offered

Package	Time of Arrival	Amount Offered
P1	1	20
P2	1.25	10
P3	1.5	30
P4	1.7	40
P5	2.0	50

Table4.2: Packages in Q2 at DC3 and their time of arrival (in the increasing order) and amount offered

Package	Time of Arrival	Amount Offered
P6	1.2	50
P7	1.5	60
P8	1.6	30
P9	1.7	45

Under condition of “Without Protocol”, truck T1 picks Q1 because it has more packages than Q2 and packages P1, P2 and P3 are loaded on the truck because of their order of their arrival at the DC.

Each simulation run was started by putting packages corresponding to one day of demand at the DCs in proportion to population around their regions, and no demand was generated after that. This was done to try to duplicate the effects of a sudden surge in demand. Based on preliminary results, each simulation was run for 50 hrs simulation time to compared results.

4.2 Input Data

Results were obtained for various ranges of package values with the following set of parameters:

1. Number of DCs: 36
2. Demand of packages per day: 10,000 and 20,000
3. Number of trucks: 36 and 72
4. Truck Capacity: 10 and 20
5. Transportation Cost: $2/mile
6. Load Factor: 0.8
7. Package Values: Derived from Beta distributions
8. Number of replications: 25

4.2.1 Mean for Beta Distribution for Package Values

Package values were derived from beta distribution in terms of $/mile. Net package value was obtained by multiplying the value obtained from beta distribution with the shortest distance between the source and destination DC for the package. Beta distribution was chosen because:

1. It gives only non-negative values, which is required for package values.
2. Using proper parameters, the shape of beta distribution can be altered to get different combinations of package values.

4.3 Results

1. **Package Values: Large Variance**

   Package values were derived from Beta (2, 5) with the mean of $3/mile. Table 4.3 and table 4.4 shows the weighted waiting time and weighted transportation time respectively obtained from simulation with and without the protocols using large variance in package values for different sets of demand, number of trucks and truck capacity.

Table4.3: Weighted Waiting Time for a PLN "Without Protocol" and "With Protocol" and percentage change in weighted waiting Time

Demand of Packages per day	Number of Trucks	Truck Capacity	Weighted Waiting Time (in hrs)		% change in Weighted Waiting Time
			With Protocol	Without Protocol	
10000	36	10	42.71	48.58	–12.08
		20	33.26	44.50	–25.26
	72	10	32.27	44.37	–27.27
		20	18.85	25.67	–26.57
20000	36	10	47.00	49.47	–4.99
		20	42.55	48.49	–12.25
	72	10	42.07	48.47	–13.20
		20	31.81	43.91	–27.56

Table4.4: Weighted Transportation Time for a PLN "Without Protocol" and "With Protocol" and percentage change in weighted transportation time

Demand of Packages per day	Number of Trucks	Truck Capacity	Weighted Transportation Time (in hrs)		% change in Weighted Transportation Time
			With Protocol	Without Protocol	
10000	36	10	44.62	48.84	–8.64
		20	37.91	45.89	–17.39
	72	10	36.98	45.84	–19.32
		20	26.55	33.61	–21.01
20000	36	10	47.74	49.54	–3.63
		20	44.52	48.78	–8.73
	72	10	44.17	48.77	–9.43
		20	36.66	45.54	–19.50

2. Package Values: Zero Variance

All the packages in the simulation have same value, i.e., $3/mile. Table 4.5 and table 4.6 shows the weighted waiting time and weighted transportation time respectively obtained from simulation with and without the protocols for demand of 20,000 packages per day with different sets of number of trucks and truck capacity.

Table4.5: Weighted Waiting Time for a PLN "Without Protocol" and "With Protocol" and percentage change in weighted waiting Time

Demand of Packages per day	Number of Trucks	Truck Capacity	Weighted Waiting Time (in hrs)		% change in Weighted Waiting Time
			With Protocol	Without Protocol	
20000	36	10	49.05	49.47	–0.85
		20	46.99	48.49	–3.09
20000	72	10	46.92	48.47	–3.20

Table4.6: Weighted Transportation Time for a PLN "Without Protocol" and "With Protocol" and percentage change in weighted transportation time

Demand of Packages per day	Number of Trucks	Truck Capacity	Weighted Transportation Time (in hrs)		% change in Weighted Transportation Time
			With Protocol	Without Protocol	
20000	36	10	49.21	49.55	–0.69
		20	47.72	48.77	–2.15
	72	10	47.66	48.76	–2.26

3. Package Values: Small Variance

Package values were derived from Beta (5, 5) with the mean of $3/mile. Tables 4.7 and 4.8 show the weighted waiting time and weighted transportation time, respectively, obtained from simulation with and without the protocols for demand of 20,000 packages per day with different sets of number of trucks and truck capacity.

Table4.7: Weighted Waiting Time for a PLN "Without Protocol" and "With Protocol" and percentage change in weighted waiting Time.

Demand of Packages per day	Number of Trucks	Truck Capacity	Weighted Waiting Time (in hrs)		% change in Weighted Waiting Time
			With Protocol	Without Protocol	
20000	36	10	48.35	49.47	–2.26
		20	45.74	48.49	–5.67
	72	10	45.55	48.48	–6.04

Table4.8: Weighted Transportation Time for a PLN "Without Protocol" and "With Protocol" and percentage change in weighted transportation time

Demand of Packages per day	Number of Trucks	Truck Capacity	Weighted Transportation Time (in hrs)		% change in Weighted Transportation Time
			With Protocol	Without Protocol	
20000	36	10	48.73	49.55	–1.65
		20	46.79	48.78	–4.08
	72	10	46.67	48.77	–4.31

Tests were performed for each set of "demand of packages per day", "number of trucks" and "truck capacity" to see statistical difference between weighted waiting times and between weighted transportation times with and without the protocols. For each set of values, 25 replications were used for simulation with the protocols and simulation without the protocols. Also, the seed used for a replication in simulation with the protocols is same as the seed used for the corresponding replication in simulation without the protocols. It was found that for each set of "demand of packages per day", "number of trucks" and "truck capacity", there is a significant difference between weighted waiting times with and without the protocols at level of significance (α) 0.05. Also, there is a significant difference between weighted transportation times, for the same set of values, with and without the protocols at the same level of significance.

From the above results, it can be seen that PLN performs better with the protocols. But difference in the performance of a PLN with and without the protocols is significantly larger with large variance in package values (in terms of $/mile) as compared to small variance and zero variance.

5. Future Work

Following issues need to be addressed to have greater insight into a PLN:

1. In order to have better understanding of the protocols mentioned in Chapter 3, the simulation needs to run from the beginning until it reaches steady state. This was not done because, currently, the simulation runs too slow to be able to generate significant results. Also, higher demand for packages per day, greater truck capacity and an increased number of trucks should be examined to better understand the system under more realistic generating conditions.
2. In the current version of simulation, the DC sends load bids when:
    a. There is generation of packages at the DC.
    b. A truck starts moving from an adjacent DC towards that DC.

    In the simulation of a PLN, an event needs to be included which will allow a DC send load bids to trucks after a specific period of time when the generation of a demand event or truck leaving from an adjacent DC to the DC might not occur for a long time. This will be especially useful to simulate peak, off-peak demand in a PLN.
3. The current version of simulation assumes the simplest behavior of a package agent. It just calculates bids for the corresponding package for various legs of its journey at the beginning of the journey. The advanced behavior of package agents needs to be explored that could affect the performance of a PLN. For example, making decision for its package concerning which truck to choose, how many hours it should wait at DC, etc.
4. The negotiation capability among various package agents, among truck agents and package and truck agents also needs to be examined to see the impact of agents' negotiations on package transport and also from the prospective of gaming system by agents.

References

[1] Kay, M.G., and Parlikad, A.N., "Material flow analysis of public logistics network," *7th Int. Material Handling Res. Colloq.*, Portland, ME, June 1–5, 2002.

[2] Chaudhary, M. L., *A first course in bulk queues*, New York: Wiley, 1983.

[3] Ortuzar, J.D., and Willumsen, L.G., *Modelling transport*, 3rd ed. Wiley, Chichester, 2001.

[4] Kay, M.G., "Protocol Design for a Public Logistics Network," in *International Material Handling Research Colloquium*, June 13–17, 2004, Graz, Austria.

[5] Joines, J.A., and Roberts, S.D., "Fundamentals of object-orientedsimulation," *Proceeding of the 1998 Winter Simulation Conference*, December 13-16, 1998, Washington DC.

[6] MacKie-Mason, J.K. and Varian H.R., "Pricing the Internet," in *Public Access to the Internet*, JFK School of Government, May 26-27, 1993.

[7] Kay, M.G., and Jain, A., "Issues in Agent-based Coordination of Public Logistics Network," *Technical Report 02-01,* Department of Industrial Engineering, North Carolina State University, Raleigh, NC, October 2002.

Appendix 1a

This appendix provides the classes used to simulate a PLN

Classes

Following are the classes defined for the simulation of Public Logistics Network (PLN):

- **Truck**

  This class represents trucks in the simulation. Following are the attributes or member variables of this class:

  Attributes or member variables

  - *truckid*
    This is a scalar quantity, which represents the truck id.
  - *sdcn*
    This is a vector. It contains the values of the source DCs i.e. the DCs from which the truck starts. So every time a truck starts from a DC, its id will be added to the vector as new element.
  - *ddcn*
    This is a vector. It contains the value of the destination DC, i.e. the DC to which the truck travels. So every time the truck moves, its destination will be added to the vector as a new element.
  - *route*
    It is a cell array. Every time the truck takes a route, a vector representing the route, will be added to the cell array.
  - *transtime*
    It is a scalar quantity. Every time the truck starts from a DC, the value is updated to the current time.
  - *maxcapacity*
    This is a scalar representing the maximum capacity of the truck.
  - *currnumpack*
    This is a scalar quantity representing the number of package the truck is carrying at any point in time.
  - *ldcn*
    This is a scalar quantity. It is same as the last DC id where the truck waited for the packages.
  - *ndcn*
    This is a scalar quantity. It is same as the next DC id to which the truck is going. If the truck is at the DC, the value is 0.
  - *cdcn*
    This is a scalar quantity. If the truck is at a DC, cdcn will have the same value as the DC id, else it is 0.
  - *ismoving*
    The value is 1, if the truck is moving else 0.
  - *packid*
    This is an array, which consists of the ids of the packages that the truck is carrying at any point in time. If the truck is waiting in the DC, the value is [ ].
  - *data*

This is an array, which will store the attributes of packages. One row of the array is dedicated for one package. There are eight columns per row as follows:

- Package id
- Starting DC or source DC id
- Destination DC id
- Number of DCs visited by the package on its journey to the destination. When the package is at source DC, the value in this column is 1.
- Total number of DCs a package has to go through in its journey (including the starting DC)
- Time a package started its journey from the starting DC
- Total time, package has spent waiting at DCs, it has gone through at any point in time.
- Time at which package is unloaded at the last intermediate DC.
- Amount offered by package for current leg of journey.
- Amount offered by package for its entire journey.
-

- *pkgcont*
  This is a cell array having information about the packages, truck has reserved at various DCs
- *dccont*
  This is a vector which stores DC ids where truck has reservations.
- *pkgnext*
  This is an array having information about the packages that have next DC as their final destinations.
- *pkgnotnext*
  This is an array having information about the packages that have next DC not as their final destinations.
- *indextonextdc*
  A vector having information about the bay number from where truck has to pick the packages, at the DCs, where truck has reservation.
- *loading*
  A flag indicates if the truck is going through the loading process. '0' means truck is not going through the loading process and '1' means truck is going through the loading process.
- *bids*
  Vector having information about the money truck would get from each DC where it has reservation.

Following are the functions in the ***Truck*** object:

- *get*
  Following are the strings passed to the function:
  - ✓ *cdcn*
    It gives the id of the DC, at which the truck is presently at. If the truck is in motion, the value is 0.
  - ✓ *currnumpack*
    It gives the number of packages, truck is currently carrying.
  - ✓ *ddcn*

It gives the id of the destination, truck is currently going to.

- ✓ *ismoving*
  It gives 0, if the truck is at a DC otherwise 1.
- ✓ *ldcn*
  It gives the id of the last dc, the truck has been to.
- ✓ *Maxcapacity*
  It gives the maximum capacity of the truck.
- ✓ *ndcn*
  It gives the id of the dc, truck is next going to.
- ✓ *packid*
  It gives the ids of the packages, the truck is carrying. If the truck is not having any package, the value is [].
- ✓ *route*
  It gives the routes the truck has gone through.
- ✓ *sdcn*
  It gives the source dc of the truck.
- ✓ *transtime*
  It gives the time at which truck begins its journey from any dc.
- ✓ *truckid*
  It gives the id of the truck, which identifies the truck uniquely.
- ✓ *data*
  It gives all the attributes of all the packages, loaded on the truck.
- ✓ *pkgcont*
  It gives information about the packages, reserved by truck.
- ✓ *dccont*
  It gives the DC idswhere truck has reservation.
- ✓ *pkgnext*
  It gives the information about the packages whose final destination is next DC.
- ✓ *pkgnotnext*
  It gives the information about the packages whose final destination is not next DC.
- ✓ *indextonextdc*
  It gives information about the bay numbers from which truck will pick the packages at different DCs
- ✓ *loading*
  It indicates if a truck is going through a loading process.
- ✓ *bids*
  It gives information about the current amount, truck will get from various DCs.

- *set*
  All the string messages used in *get* method, are used in *set* method except ‘truckid’. So at run time, the id of the truck cannot be changed.
  This method is used to set the values into the truck object.

- **DC**

  This class represents a Distribution center. Following are the attributes or member variables of this class:

  Attributes or member variables:

  - *dcid*

    This is a scalar quantity and it is the id of a DC.
  - *maxcapacity*

    This is a scalar quantity and the value is the maximum capacity of a DC.
  - *currnumpack*

    This is a scalar quantity and the value is the current number of packages in the DC at any point in time.
  - *connecteddcs*

    This is a vector, each element of which is the DC id directly connected to this DC. The first element of the vector is the id of this DC itself.

  Following are the methods in DC class:

  - *get*

    The following string messages are used as one of the parameters:
    - *connecteddcs*

      This gives a vector containing ids of all the DCs connected to the DC. The first element is the id of the DC itself.
    - *currnumpack*

      This gives the current number of packages in the DC.
    - *dcid*

      This gives the id of the DC, which identifies the DC uniquely.
    - *maxcapacity*

      This gives the maximum capacity of the DC.
  - *set*

    The following string messages are used as one of the parameters:
    - *currnumpack*

      It sets the current number of packages in the DC
    - *maxcapacity*

      It sets the maximum capacity of the DC.

- **SupplierDC**

  This class represents the supplier DC, which just supplies packages to other DCs. Following are the attributes or member variables of this class:

  Attributes or member variables:

  - *packid*

    This is a vector of all the package ids that are in this DC at a given point in time. If a package leaves the DC, the id is removed from this vector.
  - *truckid*

    This is a vector of all the truck ids that are in this DC at a given point in time. If a truck leaves the DC, the id is removed from this vector.
  - *packsuppbay*

    This vector has same number of elements as the “connecteddcs” attribute of class “dc”. Each element in the vector is the sum of values of packages in the queue going to one of the adjacent DC.
  - *numpacksuppbay*

This vector has same number of elements as the "connecteddcs" attribute of class "dc". Each element in the vector is the number of packages in the queue going to one of the adjacent DC.

Following are the methods in DC class:

- *get*
  The following string messages are used as one of the parameters:
  - ✓ *packid*
    This gives a vector containing ids of all the packages in the DC.
  - ✓ *packsuppbay*
    This gives the value of different queues. Each queue is for one DC directly connected to this DC.
  - ✓ *numpacksuppbay*
    This gives the number of packages in different queues. Each queue is for one DC directly connected to this DC.
  - ✓ *truckid*
    This gives a vector containing the id of all the trucks present at the DC.
- *set*
  All the string messages used in *get* method are used in *set* method. This function sets the value in the object.
- *removepack*
  This method removes the id of the package (from *packid* field) that has been removed from the DC because it has been loaded on the truck.
- *removetruck*
  This method removes the id of the truck from the *trucked* field (queue) in the DC because it has left the DC.

- **PlnDC**
  This class represents a DC in Public Logistics Network. This class is a child class of *dc* and *supplierdc* classes. It inherits all the field variables and methods of its super classes. Following are the attributes of this class:
  Attributes or member variables:
  - *data*
    This is an array, which will store the attributes of packages. One row of the array is dedicated for one packages. There are eight columns per row as follows:
    - Package id
    - Starting DC or source DC id
    - Destination DC id
    - Number of DCs visited by the package on its journey to the destination. When the package is at source DC, the value in this column is 1.
    - Total number of DCs a package has to go through in its journey (including the starting DC)
    - Time a package started its journey from the starting DC
    - Total time, package has spent waiting at DCs, it has gone through at any point in time.
    - Time at which package is unloaded at the last intermediate DC.

- Amount offered by package for current leg of journey.
- Amount offered by package for its entire journey.

- *infot*
  A cell array having information about the packages at and inbound a DC. Number of cells in an array is same as number of bays at a DC. It keeps information about package id, package bid and time of its arrival at the DC.
- *info*
  Same as *infot*. This is used in the bidding process in the simulation. It was used to make simulation fast.
- *incomngtrucks*
  A matrix having information about the incoming trucks and their time of arrival at a DC
- *packtrk*
  A matrix having information about packages at a DC and their associations with trucks.

Following are the methods in DC class:

- *get*
  The following string messages are used as one of the parameters:

  - *data*
    It gives all the attributes of all the packages, currently at the DC.
  - *infot*
    It gives a cell array having information about package id, package bid and time of its arrival at the DC.
  - *info*
    It gives a cell array having information about package id, package bid and time of its arrival at the DC.
  - *incomingtrucks*
    It gives information about trucks and their arrival time at a DC.
  - *packtrk*
    It gives information about packages at a DC and their association with the trucks.
- *set*
  All the string messages used in *get* method are used in *set* method. This function sets the value in the object.

- **Eventcal**
  This class maintains event calendar for the simulation.
  Following are the functions in this class:
  - *getnextevent*
    This function returns the event that is scheduled.
  - *putnextevent*
    This function schedule the occurrence of the event.

Appendix 1b

This appendix provides the MATLAB functions which were used with classes described in Appendix 1a to simulate a PLN to get an analytical model to calculate waiting time.

1. [createdc.m] is a MATLAB file which is used to create PLN DC objects in the simulation.

```
1   function c = createdc(n,IJD2,trk,W,pick)
2   %
3   %
4   %c = createdc(n,IJD2,trk,W,pick)
5   %
6   %c = an array containing plndc objects
7   %n = number of DCs in the system
8   %IJD2 = IJD matrix for a PLN
9   %trk = an array of truck objects
10  %W = link flow
11  %pick = way a queue can be picked at a DC
12
13  seq = [];
14  A = full(list2adj(IJD2));
15
16  disp('Generating DC....');
17
18  for i = 1:n
19  %get the list of connected DCs
20  conn = [i];
21  for j = 1:size(A,2)
22  if A(i,j) ~= 0
23  conn = [conn j];
24  end
25  end
26  if pick == 2
27  W1 = W(i,conn(2:end));
28  seq = fixseq(W1);
29  end
30
31  %set the number of packages in each bay as 0
32  suppbay = zeros(size(conn,1),size(conn,2));
33
34  %set the package ids as null
35  pid = [];
36
37  %set the truck ids as null
38  tid = [];
39
40  %for the DC object
41  c(i) = plndc(i,inf,0,conn,suppbay,suppbay,pid,tid,[],seq);
42
43  end
```

2. [createtrucks.m] is a MATLAB function which is used to create truck objects

```
1    function trk = createtrucks(TSI,TK)
2    %
3    %
4    %trk = createtrucks(TSI,TK)
5    %
6    %trk = array of truck objects
7    %TSI = vector containing DC ids assigned to trucks
8    %TK = structure containing truck information.
9    disp('Generating Trucks....');
10    for i = 1:length(TSI)
11    trk(i) = truck(i,TSI(i),TSI(i),{},0,TK.mxtrkcap,0,TSI(i),TSI(i),TSI(i),0,0,0,0,0);
12    end
```

3. [genAvgTransTime.m] is a MATLAB function which gives association of trucks with DCs in the beginning of the simulation.

```
1   %NT = genAvgTransTime(mct,pcd,IJUL,w,td)
2   %NT    -  Number of trucks
3   %mct   -  scalar value - maximum capacity of truck.
4   %pcd   -  scalar value - percentage demand of UPS considering.
5   %w     -  Demand matrix between DC i and DC j
6   %td    -  Total demand of the packages
7   %IJT   -  node-node-transportation time matrix
8   %lf    -  load factor
9
10  function NT = genAvgTransTime(mct,w,td,IJT,lf)
11
12  dim = size(w,1);
13
14  %Make a matrix for 24 hrs
15  H=zeros(dim,dim);
16  H(w>0)=24;
17
18  %get the number of trucks
19  NT1=((w)*td)/(mct*lf);
20  NT = (NT1.*full(list2adj(IJT)))/24;
```

4. [genrandnum.m] is a MATLAB function which generates random numbers in the simulation.

```
1   %This function generat a random number based on the "mean" provided to it
2   function ran = genrandnum(mean,type)
3   %
4   %ran = genrandnum(mean,type)
5   %ran = random number
6   %mean = mean for exponential distribution to generate random number
7   %type = string for the name of the distribution
8   r = [];
9   if type == 'exp'
10   r = exprnd(mean,1,1);
11   end
12   ran = r(1,end);
```

5. [initializesimulation.m] is a MATLAB function which initializes simulation.

```
1    function [ev,trk,pdc,packarrevent,allpackids,packid,T,routes] =
initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W)
2    %
3    %[ev,trk,pdc,packarrevent,allpackids,packid,T,routes] =
initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W)
4    %ev = object for event calendar
5    %trk = array for truck object
6    %pdc = array for plndc objects
7    %packarrevent = array contianing package information
8    %T = matrix having time between each connecting DCs
9    %routes = cell array for the shortest route between two DCs
10   %tp = demand for package for each pari of DCs
11   %IJD2 = IJD matrix for a PLN
12   %mrndn = matrix having means to generate demand for each pair of DCs
13   %TSI = vector having DC ids for different trucks at the start of the
14   %simulation
15   %SIM = values for the simulation
16   %TK = information to create truck objects
17   %pick = way a queue can be picked at a DC
18   %W = link flow
19
20   A = full(list2adj(IJD2));
21   T = A/ts;
22   dim = size(A,1);
23   packid = 0;
24
25   [d,pred] = dijk(list2adj(IJD2));
26   routes = pred2path(pred);
27   for i = 1:size(routes,1)
28   routes{i,i} = [routes{i,i} routes{i,i}];
29   end
30
31   %create the trucks
32   trk = createtrucks(TSI,TK);
33
34   %create event calendar
35   ev = eventcal;
36
37   %create DCs
38   pdc = createdc(dim,IJD2,trk,W,pick);
39
40   %For package traking in event calendar
41   packarrevent = [];
42   nt = length(trk);
43   tnow = 0;
44   disp('putting trucks arrival in the event calendar....');
45   for i = 1:nt
46   %1 for arrival of trucks at a DC
```

```
ev = putnextevent(ev,[0 i 1]);
end

%putting the formation of graph in the event calendar
ev = putnextevent(ev,[1 5 5]);
%putting the package arrival in the event calendar
allpack = [];
allpackids = [];
disp('putting packages in the event calendar....');
for i = 1:size(mrndn,1)
for j = 1:size(mrndn,2)
packid = packid + 1;
narrt = genrandnum(mrndn(i,j),'exp');
ev = putnextevent(ev,[tnow+narrt,packid,3]);
nodc = length(routes{j,i});
packarrevent = [packarrevent;packid i j 0 nodc 0 0 0];
end
end

ev = putnextevent(ev,[SIM.wp -2 -2]);
ev = putnextevent(ev,[SIM.rl -1 -1]);
[ev,r] = getnextevent(ev,0);
```

6. [initializevariables.m] is a MATLAB function which initializes variables at the beginning of the simulation.

```
1   function  [ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK)
2   %
3   %[ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK)
4   %TK = information to create truck objects
5   %DC = information to create dc objects
6   %PK = information to create package objects
7   %Truck Speed
8   ts = TK.trkspeed;
9
10  %Loading/Unloading time
11  lut = DC.lut;
12
13  %Processing time for packages
14  ppt = DC.ppt;
15
16  %break variable
17  out = 0;
18
19  %Number of observations
20  rd = 0;
21
22  %total time
23  td = 0;
24
25  %package id
26  packid = 0;
27
28  %total package demand between different DCs
29  tp = PK.tp;
30
31  %Truck distribution at different DCs
32  TSI = TK.tsi;
33
34  %percentage package distribution for each route
35  W = PK.w;
```

7. [intializerandomnumbers.m] is a MATLAB function which initializes random number generation.

```
1    %This function initialize the variables used to generate random numbers.
2    function [mrndn] = intializerandomnumbers(tp)
3    %
4    %[mrndn] = intializerandomnumbers(tp)
5    %mrndn = matrix having means for distribution to generate demand for each
6    %pair of DCs
7    %tp = demand for packages for each pair of DCs
8    %For random number
9    dim  = size(tp,1);
10    %set the mean for each DC combination
11    mrndn = (ones(dim,dim)./tp)*24;
```

8. [loadpackages.m] is a MATLAB function which simulation loading of packages on a truck.

```
1    function [trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] =
loadpackages(currdcid,ev,trk,pdc,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum)
2    %
3    %[trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] =
loadpackages(currdcid,ev,trk,pdc,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum)
4    %trk = array for truck objects
5    %pdc = array for pln dc objects
6    %ev = object for event calander
7    %W1 = link flow
8    %currdcid = current DC id
9    %tnow = current simulation time
10   %lut = loading/unlaoding time
11   %pick = way to pick pakcage queue
12   %ve = not is use
13   %routes = cellarray habing shortest distance between each pair of DCs
14   %W = link flow
15   %waitMean = runnign average of the waiting time
16   %waitVar = running variance of waiting time
17   %waitNum = number of observations to calculate current mean waiting time
18
19   %Get the current DC
20   cd = pdc(currdcid);
21
22   %get the next DC for which there is a queue with maximum value
23   ndcids = get(cd,'connecteddcs');
24   v = [];
25
26   if pick == 1          %Queue with maximum value
27   am = argmax(get(cd,'packsuppbay'));
28   ndcid = ndcids(am);
29   W1 = [];
30   dcnexts = get(cd,'numpacksuppbay');
31   dcnext = dcnexts(am);
32
33
34   elseif pick == 0
35   %With random pick
36   rndpick = 0;
37   ndcids1 = ndcids(2:end);
38   W1 = W(currdcid,ndcids1);
39   while rndpick == 0
40   ams = wtselect(W1,1,1);
41   am = ams;
42   am = am + 1;
43   ndcid = ndcids(am);
44   dcnexts = get(cd,'numpacksuppbay');
45   dcnext = dcnexts(am);
46   if dcnext ~= 0
```

```
47  rndpick = 1;
48  end
49  end
50
51  elseif pick == 2       %For scheduled truck arrival
52  seqpick = 0;
53  while seqpick == 0
54  currseq = get(cd,'currseqval');
55  cd = set(cd,'currseqval');
56  am = currseq + 1;
57  ndcid = ndcids(am);
58  dcnexts = get(cd,'numpacksuppbay');
59  dcnext = dcnexts(am);
60  if dcnext ~= 0
61  seqpick = 1;
62  end
63  end
64  W1 = [];
65  end
66
67
68  %get the first truck from the DC queue
69  trkids = get(cd,'truckid');
70  ntrkid = trkids(1);
71
72  %set the number of packages that the truck will carry
73  if dcnext > get(trk(ntrkid),'maxcapacity')
74  numpack = get(trk(ntrkid),'maxcapacity');
75  else
76  numpack = dcnext;
77  end
78
79  %get all the packages in the DC
80  pksid = get(cd,'data');
81  pksidtemp = [];
82  j = [];
83  for i = 1:size(pksid,1)
84  ndc = routes{pksid(i,3),pksid(i,2)}(pksid(i,4)+1);
85  %select the package whose next destination is same as the selected next DC
86  if ndc == ndcid
87  j = [j;i];
88  if size(j,1) == numpack
89  break;
90  end
91  else
92  pksidtemp = [pksidtemp;pksid(i,:)];
93  end
94  end
95  pksidtemp = [pksidtemp;pksid((i+1):end,:)];
96  %packages selected to be put on the truck
```

```
97    pkgleave = pksid(j,:);
98
99    if size(pkgleave,1) > 0
100   %set the statistics for the package to be shipped
101   pkgleave(:,7) = pkgleave(:,7) + (tnow - pkgleave(:,8));
102
103   %Added to calculate the statistics
104   if tnow > 300
105   cdc = currdcid;
106   ndc = ndcid;
107   for i = 1:size(pkgleave,1)
108   wait = tnow - pkgleave(i,8);
109   oldMean = waitMean(cdc,ndc);
110   waitMean(cdc,ndc) = (waitMean(cdc,ndc)*waitNum(cdc,ndc) +
wait)/(waitNum(cdc,ndc)+1);
111   if waitNum(cdc,ndc) >= 2
112   waitVar(cdc,ndc) = power((power(waitVar(cdc,ndc),2)*(waitNum(cdc,ndc)-1) +
waitNum(cdc,ndc)*power(waitMean(cdc,ndc)-oldMean,2) + power(waitMean(cdc,ndc)-
wait,2))/(waitNum(cdc,ndc)),0.5);
113   elseif waitNum(cdc,ndc) == 1
114   waitVarTemp = wait;
115   waitVar(cdc,ndc) = power(power(waitMean(cdc,ndc)-waitVarTemp,2) +
power(waitMean(cdc,ndc)-waitVar(cdc,ndc),2),0.5);
116   elseif waitNum(cdc,ndc) == 0
117   waitVar(cdc,ndc) = wait;
118   end
119   waitNum(cdc,ndc) = waitNum(cdc,ndc) + 1;
120   end
121   %End of addition
122
123   %set the variables for truck
124   trk(ntrkid) = set(trk(ntrkid),size(j,1),'currnumpack');
125   trk(ntrkid) = set(trk(ntrkid),ndcid,'ndcn');
126   trk(ntrkid) = set(trk(ntrkid),currdcid,'cdcn');
127   trk(ntrkid) = set(trk(ntrkid),pkgleave,'data');
128
129   %set the variables for DC
130   cd = set(cd,size(pkgleave,1),'currnumpack');
131   cd = set(cd,size(pkgleave,1),am,0,'packsuppbay');
132   cd = set(cd,size(pkgleave,1),am,0,'numpacksuppbay');
133
134   end
135
136   cd = set(cd,pksidtemp,'setdata');
137
138   pdc(currdcid) = cd;
139
140   %remove the truck from the queue
141   pdc(currdcid)= removetruck(pdc(currdcid),ntrkid);
142
```

```
143   %schedule the departure of the truck
144   ev = putnextevent(ev,[tnow+lut/60,ntrkid,0]);
```

9. [nextarrival.m] is a MATLAB function which simulates generation of demand.

```
1     function [packarrevent,packid,ev] =
nextarrival(packarrevent,packid,mrndn,ev,IJD2,tnow,m,routes)
2     %
3     %
4     %[packarrevent,packid,ev] =
nextarrival(packarrevent,pid,packid,mrndn,ev,IJD2,tnow,m,routes)
5     %packarrevent = an array having all the informaiton about packages
6     %packid = id of next package
7     %mrndn = an array having mean of distribution to generate demand for each pair of DCs
8     %ev= an object to maintain event calendar for simulation
9     %IJD2 = IJD matrix for a PLN
10    %tnow = current time of the simulation
11    %m = row id for the current package
12    %routes = cell array having shortest route between each pair of DCs
13
14
15    %get the source Dc and destination DC
16    custdc = packarrevent(m,2);
17    suppdc = packarrevent(m,3);
18
19    if m == size(packarrevent,1)
20    packarrevent = packarrevent(1:m-1,:);
21    elseif m == 1
22    packarrevent = packarrevent(2:end,:);
23    else
24    packarrevent = [packarrevent(1:m-1,:);packarrevent(m+1:end,:)];
25
26
27    %get the next package id
28    packid = packid + 1;
29
30    %get the time for next demand arival
31    narrt = genrandnum(mrndn(custdc,suppdc),'exp');
32    nextarrivaltime = tnow+narrt;
33
34    %Schedule the arrival of next demand
35    ev = putnextevent(ev,[tnow+narrt packid 3]);
36
37    %put the information regarding the package scheduled to be generated in
38    %a seperate matrix
39    nodc = length(routes{suppdc,custdc});
40    packarrevent = [packarrevent;packid custdc suppdc 0 nodc 0 0 0];
```

10. [pln.m] is a MATLAB function which provides information about PLN configuration.

```
1    function [tp,TSI,W] =  pln(pop,xy,IJD,TC,ts,p,lu,td,lf)
2    %
3    %[NT,AT,tp] = pln(pop,xy,IJD,TC,ts,p,lu,td)
4    %pop    -   Population matrix(36X1) for  each DC
5    %xy     -   Location matrix(36X2) for each DC
6    %IJD    -   Matrix containing the tail node, the head node and the distances between the
tail and the head nodes
7    %TC     -   n-element Truck Capacity Vector
8    %ts     -   Speed of the truck
9    %p      -   m-element Proximity factor vector, contains different possible proximity
factors
10   %lu     -   l-element loading/unloading time vector, contains different possible loading
unloading time in minutes
11   %td     -   Total demand (number of packages)
12   %lf     -   Load factor
13
14
15   %Sort the matrices supplied
16   TC = sort(TC);
17   p = sort(p);
18   lu = sort(lu);
19
20   %Generate the demand at all DCs from the population around it.
21   dem = (pop*100)/sum(pop);
22
23   % Generate the demand from DC j at DC i
24   w = dem*dem';
25   dim = size(w,1);
26   prox1{1} =  useProximity(p(1),w,xy);
27   tp = prox1{1}*td;
28
29   %Get IJT matrix for all loading and unloading time
30   for j = 1:length(lu)
31   IJT = IJD;
32   IJT(:,3) = IJD(:,3)/ts;
33   lut{j} = IJT;
34   end
35
36   %Get the demand matrix according to the proximity factor and the loading/unloading
time
37   for l = 1:length(p)
38   disp('Generating data...')
39   for q = 1:length(lu)
40   %Get all the routes
41   A = full(list2adj(lut{q}));
42   [D,P]=dijk(A);
43   rte=pred2path(P);
44
```

```
%Get the demand for each arc in the network
w3 = zeros(dim,dim);
for i=1:dim
for j=1:dim

%Considers all the possible paths from P(pred2path)
b=rte{i,j};
if i == j
b = [b b];
end

%Adds the demand of the route to the demand of the DC on the route
for k=1:length(b)-1
w3(b(k),b(k+1)) = w3(b(k),b(k+1)) + prox1{l}(i,j);
end
end
end
prox{l,q} = w3;
end
end
W = w3;
m = 0;
for y = 1:length(p)
for z = 1:length(TC)
m = m + 1;
n = 0;
for s = 1:length(lu)
n = n + 1;

%get the number of trucks
NT = genAvgTransTime(TC(z),prox{y,s},sum(sum(tp)),IJT,lf);


%get the truck distribution for different DCs based on
%the demand
TSI = wtselect(dem/100,1,ceil(sum(sum(NT))));
end
end
end
```

11. [simulation.m] is a MATLAB function which formats the input before starting the simulation.

```
1    function [AT,WT] =
simulation(nopack,trkcap,trksp,lu,pf,pt,wup,runlen,s,IJD2,xy,pop,lf,pick,tp,TSI,W)
2    %
3    %[AT,WT] =
simulation(nopack,trkcap,trksp,lu,pf,pt,wup,runlen,s,IJD2,xy,pop,lf,pick,tp,TSI,W)
4    %AT = Average transportation time from simulation
5    %WT = Average waiting time from simulation
6    %nopack = total packages in the system per day
7    %trkcap = truck capacity
8    %trksp = truck speed
9    %lu = loadin/unloading time
10   %pf = proximity factor
11   %wup = simulation warm-up period
12   %runlen = simulation run length
13   %s = seed in simulation
14   %IJD2 = IJD matrix for simulation
15   %xy = XY coordinate of each DC
16   %pop = population around each DC
17   %lf = load factor
18   %pick = way a queue of pakcages is picked
19   %tp = total package deand between each pair of DCs
20   %TSI = vector having DC id assigned to each truck
21   %W = link flow
22
23
24   TK.trkspeed = trksp;       %Truck speed
25   TK.mxtrkcap = trkcap;      %truck max. capacity
26   TK.loadfact = lf;          %load factor
27   TK.tsi = TSI;              %Truck distribution at different DCs
28
29   PK.total = nopack;         %total package in the system per day
30   PK.tp = tp;                %total packages between diffferent DCs
31   PK.w = W;                  %packages for each route
32
33   DC.XY = xy;
34   DC.Pop = pop;
35   DC.lut = lu;               %loading/unloading time at DC
36   DC.prxf = pf;              %Proximity factor
37   DC.ppt= pt;                %Process time for demand
38
39   SIM.wp = wup;              %Simulation warmup period
40   SIM.rl = runlen;           %Simulation rum length
41   SIM.seed = s;         %Seed in simulation
42   [AT,WT] = startsimulation(DC,TK,PK,SIM,IJD2,pick);
```

12. [start.m] is a MATLAB function where the input is provided for the simulation.

```
function start

%load data related to PLN having 36 DCs
load plnex36;
%truck capacity
trkcap = [20];
% number of packages per day in the system
nopack = [5000];
%truck speed
trkspeed = [60];
%loading/unloading time
lut = [0];
%proximity factor
pf = [0];
%processing time in DC
pt = 0;
%simulation warm-up period
warmupperiod = 0;
%simulation run length
runlength = 1000;
%load factor
lf = [0.8];
%1 for maximum value or number queue, 0 for weighted selection, 2 for the scheduled version
qpick = [1];

for qp = 1:length(qpick)
for p = 1:length(seed)
for i = 1:length(nopack)
for j = 1:length(trkcap)
for k = 1:length(trkspeed)
for m = 1:length(lut)
for n = 1:length(pf)
for q = 1:length(lf)
%For PLN
[tp,TSI,W] = pln(DC.Pop,DC.XY,IJD2,trkcap(j),trkspeed(k),pf(n),lut(m),nopack(i),lf(q));
%For simulation
[ATT,WT] = simulation(nopack(i),trkcap(j),trkspeed(k),lut(m),pf(n),pt,warmupperiod,runlength,seed(p),IJD2,DC.XY,DC.Pop,lf(q),qpick(qp),tp,TSI,W);
end
end
end
end
end
end
end
end
end
```

12. [startsimulation.m] is a MATLAB function which is the main simulation function.

```
1    function  [ATT,avgwt] = startsimulation(DC,TK,PK,SIM,IJD2,pick)
2    %
3    %[ATT,avgwt] = startsimulation(DC,TK,PK,SIM,IJD2,pick)
4    %ATT = Average transportation time from simulation
5    %avgwt = Average waiting time from simulation
6    %DC = Information about DCs in a PLN
7    %TK = Information about trucks in a PLN
8    %PK = Information about packages in aa PLN
9    %SIM= Information about simulation
10    %IJD2 = IJD matrix for a PLN
11    %pick = way a queue of packages is picked
12
13    %Setting the seed for the simulation run
14    rand('state',SIM.seed);
15    permseedval = SIM.seed;
16
17
18    %initialize global variables
19    [ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK);
20
21    %Initialize random numbers
22    [mrndn] = intializerandomnumbers(tp);
23
24    %Initialize other objects in simulation
25    [ev,trk,pdc,packarrevent,allpackids,packid,T,routes] =
initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W);
26
27    %For statistics collection and plotting graphs
28    si = size(tp);
29    intArrMean = zeros(si);
30    intArrVar = zeros(si);
31    intArrNum = zeros(si);
32    intArrOld = zeros(si);
33    loadMean = zeros(si);
34    loadVar = zeros(si);
35    loadNum = zeros(si);
36    waitMean = zeros(si);
37    waitVar = zeros(si);
38    waitNum = zeros(si);
39
40    out = 0;
41    count = 0;
42    totpack = 0;
43    totval = 0;
44    totval2 = 0;
45
46    avgtrkld = [0 0];
47    avgtt = [0 0];
```

```
avgdcld = [0 0];
avgwtp = [0 0];
avgwt = 0;
ATT = 0;
totearn = 0;
intcount = 0;
totval = 0;
ve = [];

% 1 - truck arrive
% 0 - truck leaving
% 3 - package arriving
% 4 - package leaving

disp('Simulation started...');
time = 1;
while(1)
[ev,rs] = getnextevent(ev,1);

%If there are no events in the event calendar
if size(rs,1) == 0
disp('No more events...');
ATT = td/rd;
break;
end
%If there are events in the event calendar, continue
r = rs(1,:);
tnow = r(1);

if r(3) == 1                %truck arriving at a DC
ev = putnextevent(ev,[tnow+lut/60,r(2),6]);
elseif r(3) == 6

%get the truck id
tr = r(2);

%Get the DC id at which truck arrives

if get(trk(tr),'cdcn') == 0
%for trucks coming from other DC
currdcid = get(trk(tr),'ndcn');
trk(tr) = set(trk(tr),currdcid,'cdcn');
trk(tr) = set(trk(tr),0,'ndcn');
else
%for the trucks coming straight to the DC. This happens once, at the beginning of the
simulation.
currdcid = get(trk(tr),'cdcn');
end

%get the packages
```

```
97    pkage = get(trk(tr),'data');
98
99    %get the number of packages in the truck
100   numpack = size(pkage,1);
101
102   if numpack > 0                %if number of packages on the truck is more than zero,
unload them
103   trk(tr) = set(trk(tr),[],'data');
104   [trk,pdc,rd,td,avgwt] =
unloadpackages(tr,currdcid,trk,pdc,rd,td,tnow,avgwt,pkage,routes);
105   end
106
107   %set the truck in the queue at a DC
108   pdc(currdcid) = set(pdc(currdcid),tr,'truckid');
109
110
111   %get the next DC for which there is a queue with maximum value
112   maxnumpack = max(get(pdc(currdcid),'packsuppbay'));
113
114   %if there are any packages lying in the DC
115   if maxnumpack ~= 0
116   [trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] =
loadpackages(currdcid,ev,trk,pdc,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum);
117   end
118
119   elseif r(3) == 0                    %truck leaving a DC
120
121   %get the truck id
122   tr = r(2);
123
124   %set the final property of the truck before its departure and
125   %put the next event of the arrival of the truck, at the next DC, in the event calendar
126
127   cdc = get(trk(tr),'cdcn');
128   ndc = get(trk(tr),'ndcn');
129
130   transtime = T(cdc,ndc);
131
132   if tnow > 4
133
134   %For Mean and variance of inter-arrival of trucks
135   oldMean = intArrMean(cdc,ndc);
136   intArr = tnow-intArrOld(cdc,ndc);
137   intArrMean(cdc,ndc) = (intArrMean(cdc,ndc)*intArrNum(cdc,ndc) + intArr
)/(intArrNum(cdc,ndc)+1);
138   if intArrNum(cdc,ndc) >= 2
139   intArrVar(cdc,ndc) = power((power(intArrVar(cdc,ndc),2)*(intArrNum(cdc,ndc)-1) +
intArrNum(cdc,ndc)*power(intArrMean(cdc,ndc)-oldMean,2) +
power(intArrMean(cdc,ndc)-intArr,2))/(intArrNum(cdc,ndc)),0.5);
140   elseif intArrNum(cdc,ndc) == 1
```

```
141   intArrVarTemp = intArr;
142   intArrVar(cdc,ndc) = power(power(intArrMean(cdc,ndc)-intArrVarTemp,2) +
power(intArrMean(cdc,ndc)-intArrVar(cdc,ndc),2),0.5);
143   elseif intArrNum(cdc,ndc) == 0
144   intArrVar(cdc,ndc) = intArr;
145   end
146   intArrNum(cdc,ndc) = intArrNum(cdc,ndc) + 1;
147
148   %For mean and variance of truck load
149   oldMean = loadMean(cdc,ndc);
150   load = get(trk(tr),'currnumpack');
151   loadMean(cdc,ndc) = (loadMean(cdc,ndc)*loadNum(cdc,ndc) +
load)/(loadNum(cdc,ndc)+1);
152   if loadNum(cdc,ndc) >= 2
153   loadVar(cdc,ndc) = power((power(loadVar(cdc,ndc),2)*(loadNum(cdc,ndc)-1) +
loadNum(cdc,ndc)*power(loadMean(cdc,ndc)-oldMean,2) + power(loadMean(cdc,ndc)-
load,2))/(loadNum(cdc,ndc)),0.5);
154   elseif loadNum(cdc,ndc) == 1
155   loadVarTemp = load;
156   loadVar(cdc,ndc) = power(power(loadMean(cdc,ndc)-loadVarTemp,2) +
power(loadMean(cdc,ndc)-loadVar(cdc,ndc),2),0.5);
157   elseif loadNum(cdc,ndc) == 0
158   loadVar(cdc,ndc) = load;
159   end
160   loadNum(cdc,ndc) = loadNum(cdc,ndc) + 1;
161   end
162
163   intArrOld(cdc,ndc) = tnow;
164
165   trk(tr) = set(trk(tr),0,'cdcn');
166   ev = putnextevent(ev,[tnow+transtime,tr,1]);
167   %processed package arriving
168   elseif r(3) == 3
169   ev = putnextevent(ev,[tnow,r(2),4]);
170   %package demand arriving
171   elseif r(3) == 4
172
173   %get the package id
174   pid = r(2);
175
176   allpack = packarrevent(:,1);
177   allpack(allpack ~= pid) = 0;
178   m = find(allpack);
179
180   %get the source DC and destination DC
181   currdc = packarrevent(m,3);
182   destdc = packarrevent(m,2);
183
184   if currdc == destdc
185   rd = rd + 1;        %to satisfy local demand
```

```
186   else
187
188   packarrevent(m,4) = 1;
189   packarrevent(m,6) = r(1);
190   packarrevent(m,7) = 0;
191   packarrevent(m,8) = r(1);
192
193   %check if trucks are available at the DC
194   numoftrucksids = get(pdc(currdc),'truckid');
195
196   ndc = routes{currdc,destdc}(2);
197   if pid == 4000
198   fprintf('coming out..\n');
199   routes{currdc,destdc}
200   end
201   %if no trucks are available at the DC
202   if size(numoftrucksids,2) == 0
203
204
205   %put the package in a queue in the DC
206   pdc(currdc) = set(pdc(currdc),packarrevent(m,:),'data');
207   pdc(currdc) = set(pdc(currdc),get(pdc(currdc),'currnumpack')+1,'currnumpack');
208   dcs = get(pdc(currdc),'connecteddcs');
209   dcst = dcs;
210   dcst(dcst ~= ndc) = 0;
211   j = find(dcst);
212
213   pdc(currdc) = set(pdc(currdc),1,j,1,'numpacksuppbay');
214   pdc(currdc) = set(pdc(currdc),1,j,1,'packsuppbay');
215   %if truck is available
216   else
217   %put the package in the truck and put the event of the departure of the truck in the
event calendar.
218   truc = numoftrucksids(1);
219   trk(truc) = set(trk(truc),packarrevent(m,:),'data');
220   trk(truc) = set(trk(truc),1,'currnumpack');
221   trk(truc) = set(trk(truc),ndc,'ndcn');
222   pdc(currdc)= removetruck(pdc(currdc),truc);
223   ev = putnextevent(ev,[tnow,truc,0]);
224   end
225   end
226
227   %Schedule the arrival of the next package for the same DC combination
228   [packarrevent,packid,ev] =
nextarrival(packarrevent,packid,mrndn,ev,IJD2,tnow,m,routes);
229
230   elseif r(3) == 5    %draw the graph
231   avgload = 0;
232   avgdcload = 0;
233   avgwttime = 0;
```

```
234  for j = 1:length(trk)
235  avgload = avgload + get(trk(j),'currnumpack');
236  end
237
238  for km = 1:length(pdc)
239  avgdcload = avgdcload + get(pdc(km),'currnumpack');
240  end
241  tnow = r(1)
242  avgwttime = avgwt/rd
243  ATT = td/rd
244
245  ev = putnextevent(ev,[r(1)+1 5 5]);
246  elseif r(3) == -1   %end of simulation
247  break;
248
249  elseif r(3) == -2   %Warm up period ends
250  td = 0;
251  rd = 0;
252  avgwt = 0;
253  end
254
255  if out == 1
256  break;
257  end
258  end
```

13. [unloadpackages.m] is a MATLAB function which simulates unloading operation.

```
1    function [trk,pdc,rd,td,avgwt] =
unloadpackages(tr,currdcid,trk,pdc,rd,td,tnow,avgwt,pkage,routes)
2    %
3    %[trk,pdc,rd,td,avgwt] =
unloadpackages(tr,currdcid,trk,pdc,rd,td,tnow,avgwt,pkage,routes)
4    %trk = array of truck objects
5    %pdc = array of plndc objects
6    %rd = number of packages reached their destinations
7    %td = current average transportation time
8    %avgwt = current average waiting time
9    %tr = truck id
10   %currdcid = current DC id
11   %pkage = array containing information about the packages being unloaded
12   %routes = cell array having shortest distance between each pair of DCs
13
14   %increase the number of intermediate DC by 1
15   pkage(:,4) = pkage(:,4) + 1;
16
17   %get the packages whose final desination is not this DC
18   pktemp = pkage(:,[4 5]);
19   pktemp(pktemp(:,1) == pktemp(:,2)) = 0;
20   mnr = find(pktemp(:,1));
21   mnrpkg = pkage(mnr,:);
22
23   %get the packages whose final desination is this DC
24   pktemp = pkage(:,[4 5]);
25   pktemp(pktemp(:,1) ~= pktemp(:,2)) = 0;
26   mr = find(pktemp(:,1));
27   mrpkg = pkage(mr,:);
28
29   if size(mr,1) > 0       %package reached the destination
30   %increase the total number of packages reached the destination by 1
31   rd = rd + size(mrpkg,1);
32   %set the new total transportation time
33   td = td + size(mrpkg,1)*tnow - sum(mrpkg(:,6));
34   %set the average waiting time
35   avgwt = avgwt + sum(mrpkg(:,7));
36   end
37   %If this is the intermediate DC
38   if size(mnr,1) > 0
39   mnrpkg(:,8) = tnow;
40   %set the pakcages in the DC
41   pdc(currdcid) = set(pdc(currdcid),mnrpkg,'data');
42   %set the variables for DC
43   pdc(currdcid) =
set(pdc(currdcid),get(pdc(currdcid),'currnumpack')+size(mnr,1),'currnumpack');
44   end
45
```

```
46  dcs = get(pdc(currdcid),'connecteddcs');
47  %increae the number of packages in the respective bays where packages are stored to be
48  %transported to their next DCs
49  for i = 1:size(mnr,1)
50  ndc = routes{mnrpkg(i,3),mnrpkg(i,2)}(mnrpkg(i,4)+1);
51  dcst = dcs;
52  dcst(dcst ~= ndc) = 0;
53  j = find(dcst);
54  pdc(currdcid) = set(pdc(currdcid),1,j,1,'numpacksuppbay');
55  pdc(currdcid) = set(pdc(currdcid),1,j,1,'packsuppbay');
56  end
57  %set the number of packages in the truck to 0
58  trk(tr) = set(trk(tr),0,'currnumpack');
```

14. [useProximity.m] is a MATLAB function which helps in demand generation based on proximity factor.

```
1   function w = useProximity(p,w,xy)
2   %
3   %w = useProximity(p,w,xy)
4   %p  -   Proximity Factor
5   %w  -   Initial weight matrix for demand.
6   %xy -   DC location matrix.
7   w2 = w;
8   dis = dists(xy,xy,'mi');
9   [y,in] = sort(dis,2);
10   for j = 1:size(w,1)
11   for k = 1:size(w,1)
12   w(j,k) = w2(j,in(j,k));
13   end
14   end
15
16   %Compute the denominator
17   td = 0;
18   m = size(w,1);
19   for k = 1:m
20   td = td + power(1-(p/m),k-1);
21   end
22   td = td/m;
23
24   %Get the numerator
25   for i = 1:m
26   for j= 1:m
27   w1(i,j) = w(i,j)*power(1-(p/m),j-1);
28   end
29   end
30   w1 = w1/td;
31   wp = w1/sum(sum(w1));
32
33   for j = 1:size(w,1)
34   for k = 1:size(w,1)
35   w(j,in(j,k)) = wp(j,k);
36   end
37   end
```

Appendix 1c

This appendix provides the MATLAB functions which were used with classes described in Appendix 1a to simulate a PLN to implement the protocols.

1. [bidding.m] is a MATLAB function which is used to simulate bidding process in the simulation.

```
1    function [furdc,tr,out,nstepn] = bidding(currdcid,routes,T,tnow,varargin)
2    %
3    %[furdc,tr,out,nstepn] = bidding(currdcid,routes,T,tnow,varargin)
4    %furdc = vector having dc ids where load formation changes because of reneging
5    %tr = truck id
6    %out = cell array having Information about the pakacages that are in loads
7    %nstepn =    contianing dc ids where load formation changes because of reneging by a
truck
8    %currdcid = current dc id
9    %routes = cell array having shortest route for each pair of DCs
10   %T = Travel time between each directly connected DCs
11   %tnow = current simulation time
12   %varargin = parameter list
13   % varargin{1} = iteration number
14   % varargin{2} = not used currently
15   global trkinfo;
16   global pdc;
17   global trk;
18
19   out = {};
20   outr = {};
21   outt = {};
22   selected = 0;
23   furdc = [];
24   tr = [];
25   nstepn = {};
26   e = 0;
27
28
29   %Get the current DC
30   cd = pdc(currdcid);
31
32   %get the trucks in the increasing order of distance from the DC
33   iter = varargin{1};
34
35   if iter == 1
36   ns = varargin{2};
37   end
38
39   if iter == 1
40   trkinfotemp = trkinfo(:,2);
41   trkinfotemp(trkinfotemp == currdcid) = 0;
42   temptime = trkinfo(:,1);
43   temptime(temptime < tnow) = tnow;
```

```
44  temptime = temptime + dijk(T,trkinfo(:,2),currdcid);
45  temp = [temptime trkinfotemp trkinfo(:,6)];
46  [t,idx] = sortrows(temp);
47
48  for i = 1:length(pdc)
49  pdc(i)= set(pdc(i),'info');
50  end
51
52
53  % %For without bidding
54  %           trkinfotemp = trkinfo(:,2);
55  %           trkinfotemp(trkinfotemp == currdcid) = 0;
56  %           temptime = trkinfo(:,1);
57  %           temptime(temptime < tnow) = tnow;
58  %           temptime = temptime + dijk(T,trkinfo(:,2),currdcid);
59  %           temp = [temptime trkinfotemp trkinfo(:,6)];
60  %
61  %           %For without protocol
62  %           trktimetemp = temp(:,1);
63  %           trktimetemp(trktimetemp - tnow > 0.00001) = 0;
64  %           trktimetempin = find(trktimetemp);
65  %           temp = temp(trktimetempin,:);
66  %           temp = [temp trktimetempin];
67  %           trkdctemp = temp(:,2);
68  %           trkdctemp(trkdctemp == 0) = -1;
69  %           trkdctemp(trkdctemp ~= 0 & trkdctemp ~= -1) = 0;
70  %           trkdctempin = find(trkdctemp);
71  %           temp = temp(trkdctempin,:);
72  %           %End of addition
73  %           [t,idx] = sortrows(temp);
74  %
75  %           %For without protocol
76  % %             currdcid
77  % %             trkinfo
78  %           idx = t(:,4);
79  % %           idx
80  %           %end of addition
81  % % End of addition
82  %
83  elseif iter ~= 1
84  idx = varargin{2};
85  conndc = varargin{3};
86  end
87
88  for k = 1:length(idx)
89
90
91  out={};
92  nstepn = {};
93  if iter == 1
```

```
94  %get the load information from the current DC
95  [outt,don,nstept,e] = biddingmult(currdcid,routes,T,tnow,idx(k),ns);
96
97  if size(outt,1) == 0
98  break;
99  end
100  pkgsm = outt{1,4};
101  if size(pkgsm,1) == 0
102  continue;
103  end
104  if don == 1
105  break;
106  elseif don == 2
107  continue;
108  end
109  else
110  %Get load information from intermediate DCs along the intended path to the current
DC
111  outt = biddingsingle(currdcid,routes,T,tnow,iter,idx(k),conndc);
112  end
113  bidtruck = idx(k);
114  dccont = get(trk(idx(k)),'dccont');
115  currdcid;
116  r = routes{trkinfo(idx(k),2),currdcid};
117  if r(end-1) ~= r(end)
118  if size(outt,1) > 0
119  if outt{1,1} ~= 0
120  [furdc,tr,outr] = bidding(r(end-1),routes,T,tnow,iter+1,idx(k),currdcid);
121  end
122  end
123  end
124  out = [outr;outt;out];
125  if iter == 1
126  if size(out,1) > 0
127  %If there is a load, see if it results in reneging. if yes, change the configuration of
128  %the affected trucks and DCs
129  [furdc,tr,done,nstepn,ind,cvimp] = changeConfig(out,tnow,T);
130  if e == 1
131  if 0 - cvimp > 0.00001
132  break;
133  end
134  end
135  end
136  if done == 1
137  if ind ~= 0
138  if size(nstept,1) > 0
139  nstepn{ind,4} = [nstepn{ind,4};nstept{1,4}];
140  end
141  else
142  nstepn = [nstepn;nstept];
```

```
143   end
144   break;
145   else
146   outr = {};
147   outt = {};
148   end
149   end
150   end
```

2. [biddingmult.m] is a MATLAB function which simulates bidding at the current DC

```
function [out,don,nstept,e] = biddingmult(currdcid,routes,T,tnow,tr,ns)
%
%[out,don,nstept,e] = biddingmult(currdcid,routes,T,tnow,tr,ns)
%out = cell array having information about the packages in loads at current DC
%don = flag to check if there are packages in the DC
%nstepn = cell array having information about the packages in the load which helps in
%further iterations to form next load
%e = not used currently
%currdcid = current DC id
%routes = cell array having shortest route between each pair of DCs
%T = Matrix having journey time between two diretly connected DCs
%tnow = current simulation time
%tr = truck id
%ns = not used currently
global trkinfo;
global pdc;
global trk;


nstept = {};
e = 0;


%Get the current DC and the information of packages (at the DC and incoming)
cd = pdc(currdcid);
dcinfo = get(cd,'info');
dcinfosize = size(dcinfo,2);
dcinfotwo = cell(1,dcinfosize);
out = {};
don = 0;
con = 0;
firsttime = [];
y = 0;

%If there are no packages at a DC
if isequal(dcinfo,dcinfotwo) == 0


t = tr;
idx = tr;


ndcids = get(cd,'connecteddcs');

packsnew = [];
packsold = [];
dcidnew = 0;
dcidold = 0;
trkid = 0;
```

```
50  bidtemp = [];
51  nopp = 0;
52  totbidfinal = 0;
53  selected = 1;
54  m = 0;
55
56  %Get the times at which different trucks are reaching at the current DC
57  times = get(cd,'incomingtrucks');
58  if size(times,1) == 0
59  times = tnow;
60  else
61  times = [tnow;times(:,2)];
62  end
63
64  %If a truck is already at the DC, make its time of arrival as the current simulation time
65  times((tnow-times) > 0.00001) = tnow;
66  times = unique(times);
67
68
69  ndci = 1:length(ndcids);
70
71  for k = 1:length(t)
72  trkcap = get(trk(idx(k)),'maxcapacity');
73
74  indexndc = 0;
75  fl = 0;
76  timepack = 0;
77
78  dcconts = get(trk(idx(k)),'dccont');
79  [dccont,mem] = ismember(dcconts,currdcid);
80  dccontin = find(dccont);
81
82  if length(dccontin) ~= 0
83  y = dccontin;
84  end
85
86  %time for truck to reach the current DC
87  timetake = dijk(T,trkinfo(idx(k),2),currdcid);
88  trktime = trkinfo(idx(k),1);
89  trktime(tnow-trktime > 0.00001) = tnow;
90  tottimetake = trktime+timetake;
91
92  if y == 0
93  con = 1;
94  else
95  con = 2;
96
97  %If truck has reservation at the current DC, time at which truck would pick the load
98  if y < length(dccont)
99  firsttime = tottimetake;
```

```
100  elseif y == length(dccont)
101  firsttime = trkinfo(tr,8);
102  end
103  end
104
105
106  %If truck has reservation at the current DC, put those packages for load formation
107  if length(dccontin) > 0
108  indextondc = get(trk(idx(k)),'indextonextdc');
109  indexndc = indextondc(dccontin);
110
111  if selected == 1
112  dcinfo = get(cd,'info');
113  if indexndc ~= 0
114  pkgconts = get(trk(tr(k)),'pkgcont');
115  end
116
117  %form cell arrays and matrix to store information about load formation
118  numofindex = zeros(length(times),length(ndci));
119  index = cell(length(times),length(ndci));
120  bids = cell(length(times),length(ndci));
121  totbid = zeros(length(times),length(ndci));
122  indexfinal = cell(length(times),length(ndci));
123  bidsfinal = cell(length(times),length(ndci));
124  pkgtime =  cell(length(times),length(ndci));
125
126  %Form various loads
127  for i = 1:length(times)
128  for jk = 1:length(ndci)
129  j = ndci(jk);
130  if size(dcinfo{j},1) > 0
131  dcinfot = dcinfo{j};
132  time = dcinfot(:,3);
133  if abs(times(i) - tnow) < 0.00001
134  time(time - tnow > 0.00001) = 0;
135  elseif tnow - times(i) > 0.00001
136  time(time - tnow > 0.00001) = 0;
137  else
138  time(time - times(i) > 0.00001) = 0;
139  end
140  in = find(time);
141
142  bids{i,jk} = dcinfot(in,2);
143  [bids{i,jk},ind] = sort(bids{i,jk});
144  index{i,jk} = in(ind);
145  pkgtime{i,jk} = dcinfot(index{i,jk},3);
146
147
%%%
```

```
148 %                 fb = bids{i,j-1};
149 %                 ft = pkgtime{i,j-1};
150 %                 fi = index{i,j-1};
151 %                 fb = bids{i,jk};
152 %                 ft = pkgtime{i,jk};
153 %                 fi = index{i,jk};
154 %
155 %                  if length(fb) > 0
156 %                    [ub,ui] = unique(fb);
157 %                     if length(ub) ~= length(fb)
158 %                         iin = 1;
159 %                         fin = 1;
160 %                         for u = 1:length(ui)
161 %                             fin = ui(u);
162 %                             [ftt,fit] = sort(ft([iin:fin]));
163 %                             ftt(end:-1:1) = ftt(1:end);
164 %                             fit(end:-1:1) = fit(1:end);
165 %                             fitt = fi([iin:fin]);
166 %                             fitt = fitt(fit);
167 %                             ft(iin:fin) = ftt;
168 %                             fi(iin:fin) = fitt;
169 %                             iin = ui(u)+1;
170 %                         end
171 %                     end
172 %                     index{i,jk} = fi;
173 %                     pkgtime{i,jk} = ft;
174 %                 end
175
%%%
```

```
196  else
197  totbid(i,j) = 0;
198  end
199  end
200  end
201  end
202  [bidmax,bidind] = max(totbid,[],2);
203  end
204  ttr = tottimetake;
205  if ttr - times(end) > 0.00001
206  e = 1;
207  else
208  e= 0;
209  end
210  timet = times;
211  if fl ~= 1
212  timet((timet - ttr) > 0.00001) = 0;
213  ind = find(timet);
214  if length(ind) == 0
215  ind = 1;
216  else
217  ind = ind(end);
218  end
219  else
220
221  end
222  out = {};
223  out = cell(1,10);
224  m = m + 1;
225  out{m,1} = idx(k);
226  out{m,3} = currdcid;
227  out{m,5} = [];
228  out{m,6} = [];
229  out{m,7} = [];
230  out{m,10} = [];
231  out{m,11} = firsttime;
232
233  mn = 0;
234  for k = ind:length(times)
235  out2t = [];
236  out8t = [];
237  out9t = [];
238  packarray = {};
239  for index = 1:size(totbid,2)
240
241  out2t = [out2t ndci(index)];
242  finalindex = indexfinal{k,index};
243
244  if size(dcinfo{1,ndci(index)},1) > 0
245  dcinfofi = dcinfo{1,ndci(index)};
```

```
246  dcinfofi2 = dcinfofi(finalindex,:);
247  packarray{1,index} = dcinfofi2;
248  else
249  packarray{1,index} = [];
250  end
251  out8t = [out8t totbid(k,index)];
252  end
253  out{m,2} = [out{m,2};out2t];
254  out{m,8} = [out{m,8};out8t];
255  out{m,4}{k-mn-ind+1,1} = packarray;
256  if fl ~= 1
257  if ttr - times(k) > 0.00001
258  out{m,10} = [out{m,10} ttr];
259  else
260  out{m,10} = [out{m,10} times(k)];
261  end
262  else
263  out{m,10} = [out{m,10} times(k)];
264  end
265  end
266
267  %If truck has reservation with the current DC, put a flag in the output
268  %of this function
269  if currdcid == dcconts(dccontin)
270  out{m,9} = [dccontin indextondc(dccontin)];
271  end
272  selected = 1;
273  end
274  pdc(currdcid) = cd;
275  else
276  don = 1;
277  end
```

3. [biddingsingle.m] is a MATLAB function which simulates bidding in the intermediate DCs.

```
1    function out = biddingsingle(currdcid,routes,T,tnow,iter,tr,conndc)
2    %
3    %out = biddingsingle(currdcid,routes,T,tnow,iter,tr,conndc)
4    %out = Informaiton about the load
5    %currdcid = current DC id
6    %routes = cell array having shortest route for each pair of DCs
7    %T = Travel time between each directly connected DCs
8    %tnow = current simulation time
9    %iter = iteration number
10   %tr = truck id
11   %conndc = list of directly connected DCs
12   global pdc;
13   global trk;
14   global trkinfo;
15
16   index = [];
17   bids = [];
18   bidsfinal = 0;
19   indexfinal = [];
20   out = {};
21   indexndc = 0;
22   fl = 0;
23
24   %get object of current DC
25   cd = pdc(currdcid);
26
27   %get the index to next dc id
28   cdcs = get(cd,'connecteddcs');
29   ndcid = conndc;
30   nd = cdcs;
31   nd(nd ~= ndcid) = 0;
32   ndin = find(nd);
33
34   %get the time taken by truck to reach the DC
35   timetake = dijk(T,trkinfo(tr,2),currdcid);
36   currtime = trkinfo(tr,1);
37   if  tnow - currtime > 0.00001
38   currtime = tnow;
39   end
40   tottimetake = currtime + timetake;
41
42   %check if truck already has reservation with the DC
43   dcconts = get(trk(tr),'dccont');
44   dccont = ismember(dcconts,currdcid);
45   dccontin = find(dccont);
46
47   %If truck has reservation with the DC, put the package it has reserved in load formation
48   if length(dccontin) > 0
```

```
indextondc = get(trk(tr),'indextonextdc');
indexndc = indextondc(dccontin);
if indexndc == ndin
if dccontin(1) == length(dcconts)
if trkinfo(tr,8)-tottimetake > 0.0001
fl = 1;
end
end
else
fl = 1;
end
else
fl = 1;
end

%If truck already has reservation and if truck reaches DC at the time for
%which it reserved packages
if fl == 0
out{1} = 0;
out{2} = 0;
out{3} = currdcid;
out{4} = [];
out{5} = [];
out{6} = [];
out{7} = [];
out{8} = 0;
out{9} = [];

%If truck does not have reservation at the DC or if truck reservation time changes
else
dcinfo = get(cd,'info');

if length(dccontin) > 0
if dccontin(1) > 0
pkgconts = get(trk(tr),'pkgcont');
end
end

%if there are packages at DC
if size(dcinfo{ndin},1) ~= 0
pkginfo = dcinfo{ndin};
time = pkginfo(:,3);
time((time - tottimetake) > 0.00001) = 0;
in = find(time);
bids = dcinfo{ndin}(in,2);
[bids,i] = sort(bids);
index = in(i);
trkcap = get(trk(tr),'maxcapacity');
end

```

```
99   b = bids;
100  if length(b) > 0
101  if length(b) > trkcap
102  bidsfinal = [sum(b(end-trkcap+1:end))];
103  indexfinal = index(end-trkcap+1:end);
104  else
105  bidsfinal = [sum(b(1:end))];
106  indexfinal = index(1:end);
107  end
108  totbid = bidsfinal;
109  out{1} = tr;
110  out{2} = ndin;
111  out{3} = currdcid;
112  out{4} = dcinfo{ndin}(indexfinal,:);
113  out{5} = [];
114  out{6} = [];
115  out{7} = [];
116  out{8} = totbid;
117  out{9} = [];
118
119  %if there are no packages at the DC
120  else
121  out{1} = tr;
122  out{2} = ndin;
123  out{3} = currdcid;
124  out{4} = [];
125  out{5} = [];
126  out{6} = [];
127  out{7} = [];
128  out{8} = 0;
129  out{9} = [];
130  end
131  end
132  out{10} = [];
133  out{11} = [];
```

4. [createdc.m] is a MATLAB function which creates PLN DC objects.

```
1   function c = createdc(n,IJD2,trk,W,pick)
2   %
3   %c = createdc(n,IJD2,trk,W,pick)
4   %c = array of plndc object
5   %n = number of DCs in a PLN
6   %IJD2 = IJD matrix for a PLN
7   %trk = a vector of truck objects
8   %W = line flow matrix
9   %pick = flag to pick a queue of packages
10  seq = [];
11  A = full(list2adj(IJD2));
12  disp('Generating DC....');
13  for i = 1:n
14
15  %get the list of connected DCs
16  conn = [i];
17  for j = 1:size(A,2)
18  if A(i,j) ~= 0
19  conn = [conn j];
20  end
21  end
22  if pick == 2
23  W1 = W(i,conn(2:end));
24  seq = fixseq(W1);
25  end
26
27  %set the number of packages in each bay as 0
28  suppbay = zeros(size(conn,1),size(conn,2));
29
30  %set the package ids as null
31  pid = [];
32
33  %set the truck ids as null
34  tid = [];
35
36  %for the DC object
37  c(i) = plndc(i,inf,0,conn,suppbay,suppbay,pid,tid,[],seq,{},{},[],0);
38  end
```

5. [createtrucks.m] is a MATLAB function which creates truck objects.

```
1    function trk = createtrucks(TSI,TK)
2    %
3    %trk = createtrucks(TSI,TK)
4    %trk = a vector of truck object
5    %TSI = a vector having dc ids assigned to each truck at the beginning of the simulation
6    %TK = information about the truck objects
7    disp('Generating Trucks....');
8    for i = 1:length(TSI)
9    trk(i) =
truck(i,TSI(i),TSI(i),{},0,TK.mxtrkcap,0,TSI(i),TSI(i),TSI(i),0,0,0,0,0,TK.trkamt);
10   end
```

6. [genAvgTransTime.m] is a MATLAB function which gives association of trucks with DCs in the beginning of the simulation.

```
1   %NT = genAvgTransTime(mct,pcd,IJUL,w,td)
2   %NT     -   Number of trucks
3   %mct    -   scalar value - maximum capacity of truck.
4   %pcd    -   scalar value - percentage demand of UPS considering.
5   %w      -   Demand matrix between DC i and DC j
6   %td     -   Total demand of the packages
7   %IJT    -   node-node-transportation time matrix
8   %lf     -   load factor
9
10  function NT = genAvgTransTime(mct,w,td,IJT,lf)
11
12  dim = size(w,1);
13
14  %Make a matrix for 24 hrs
15  H=zeros(dim,dim);
16  H(w>0)=24;
17
18  %get the number of trucks
19  NT1=((w)*td)/(mct*lf);
20  NT = (NT1.*full(list2adj(IJT)))/24;
```

7. [genrandnum.m] is a MATLAB function which generates random numbers in the simulation.

```
1   function [ran] = genrandnum(varargin)
2   %
3   %[ran] = genrandnum(varargin)
4   %ran = random number
5   %varargin = list of parametes
6   %   varargin{1} = mean of a distribution
7   %   varargin{2} = name of the distribution
8   %   varargin{3} = a flag to generate demand based on time
9   r = [];
10   if length(varargin) == 2
11   switch varargin{2}
12   %For exponential distribution
13   case 'exp'
14   r = exprnd(varargin{1},1,1);
15   %For unifrom distribution
16   case 'unif'
17   r = unifrnd(0,50,1,1)*20;
18   end
19   elseif length(varargin) == 3
20   fl = varargin{3};
21   switch varargin{2}
22   %For time based uniform distribution
23   case 'unif'
24   if fl == 1
25   r = unifrnd(0,50,1,1)*20;
26   elseif fl == 2
27   r = unifrnd(0,50,1,1)*20*5;
28   elseif fl == 3
29   r = unifrnd(0,50,1,1)*20/5;
30   end
31   end
32   end
33   ran = r(1,end);
```

8. [initializesimulation.m] is a MATLAB function which initializes the simulation.

```
1    function [ev,trk,pdc,packarrevent,allpackids,packid,T,routes,trkinfo] =
initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W,PK,B)
2    %
3    %[ev,trk,pdc,packarrevent,allpackids,packid,T,routes] =
initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W)
4    %ev = object for event calendar
5    %trk = array for truck object
6    %pdc = array for plndc objects
7    %packarrevent = array contianing package information
8    %T = matrix having time between each connecting DCs
9    %routes = cell array for the shortest route between two DCs
10   %tp = demand for package for each pari of DCs
11   %IJD2 = IJD matrix for a PLN
12   %mrndn = matrix having means to generate demand for each pair of DCs
13   %TSI = vector having DC ids for different trucks at the start of the
14   %simulation
15   %SIM = values for the simulation
16   %TK = information to create truck objects
17   %pick = way a queue can be picked at a DC
18   %W = link flow
19   %PK = Information about packages
20   %B = vector having package values
21   global pdc;
22   A = full(list2adj(IJD2));
23   T = A/ts;
24   dim = size(A,1);
25   packid = 0;
26
27   %Matrix having number of packages for each pair of DCs
28   wa = PK.ar/10000;
29   wa = ceil(wa*PK.total);
30
31   [d,pred] = dijk(list2adj(IJD2));
32   routes = pred2path(pred);
33   for i = 1:size(routes,1)
34   routes{i,i} = [routes{i,i} routes{i,i}];
35   end
36
37   %create the trucks
38   trk = createtrucks(TSI,TK);
39
40   %create event calendar
41   ev = eventcal;
42
43   %create DCs
44   pdc = createdc(dim,IJD2,trk,W,pick);
45
46   %For package traking in event calendar
```

```
47  packarrevent = [];
48  nt = length(trk);
49  tnow = 0;
50  disp('putting trucks arrival in the event calendar....');
51  for i = 1:nt
52  %1 for arrival of trucks at a DC
53  ev = putnextevent(ev,[0.1 i 1]);
54  end
55
56  %putting the formation of graph in the event calendar
57  ev = putnextevent(ev,[1 5 5]);
58
59  %putting the package arrival in the event calendar
60  allpack = [];
61  allpackids = [];
62  disp('putting packages in the event calendar....');
63  %for i = 1:size(mrndn,1)
64  %for j = 1:size(mrndn,2)
65  %packid = packid + 1;
66  %narrt = genrandnum(1/mrndn{i,j}(1),'exp');
67  %ev = putnextevent(ev,[tnow+narrt,packid,3]);
68  %nodc = length(routes{j,i});
69  %packarrevent = [packarrevent;packid i j 0 nodc 0 0 0 0 0 0 0];
70  %end
71  %end
72  BV = B;
73  for i = 1:size(wa,1)
74  for j = 1:size(wa,2)
75  if i ~= j
76  for k = 1:wa(i,j)
77  packarr = [];
78  packid = packid + 1;
79
80  %get the value of package
81  bidamt = BV(packid);
82
83  %Put the packages under 'info' for DC
84  dis = dijk(A,j,i);
85  bidamt = bidamt*dis;
86  ndc = routes{j,i}(2);
87  condc = get(pdc(j),'connecteddcs');
88  [x,y] = ismember(ndc,condc);
89
90  %make packarrevent for the packages
91  nodc = length(routes{j,i});
92  packarr = [packid i j 1 nodc 0 0 0 0 bidamt 0 0];
93  packarr(9) = packarr(10)*A(j,ndc)/dijk(A,packarr(3),packarr(2));
94  pdc(j) = set(pdc(j),[packid packarr(9) 0.05],y,'infot');
95  pdc(j) = set(pdc(j),packarr,'data');
96  pdc(j) = set(pdc(j),[packid 0],'packtrk');
```

```
97   end
98   end
99   end
100   end
101
102   ev = putnextevent(ev,[SIM.wp -2 -2]);
103   ev = putnextevent(ev,[SIM.rl -1 -1]);
104   [ev,r] = getnextevent(ev,0);
105
106   %Add for truck location information
107   trkinfo = zeros(length(trk),11);
108   trkinfo(:,4) = TK.trkamt;
109   for i = 1:length(trk)
110   trkinfo(i,2) = get(trk(i),'cdcn');
111   pdc(trkinfo(i,2)) = set(pdc(trkinfo(i,2)),[i 0],'incomingtrucks');
112   end
```

9. [initializevariables.m] is a MATLAB function which initializes the variables in the simulation.

```
1   function  [ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK)
2   %
3   %[ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK)
4   %TK = information to create truck objects
5   %DC = information to create dc objects
6   %PK = information to create package objects
7   %Truck Speed
8   ts = TK.trkspeed;
9
10  %Loading/Unloading time
11  lut = DC.lut;
12
13  %Processing time for packages
14  ppt = DC.ppt;
15
16  %break variable
17  out = 0;
18
19  %Number of observations
20  rd = 0;
21
22  %total time
23  td = 0;
24
25  %package id
26  packid = 0;
27
28  %total package demand between different DCs
29  tp = PK.tp;
30
31  %Truck distribution at different DCs
32  TSI = TK.tsi;
33
34  %percentage package distribution for each route
35  W = PK.w;
```

10. [intializerandomnumbers.m] is a MATLAB function which initializes the random numbers in the simulation.

```
1    %This function initialize the variables used to generate random numbers.
2    function [mrndn] = intializerandomnumbers(tp)
3    %
4    %[mrndn] = intializerandomnumbers(tp)
5    %mrndn = matrix having means for distribution to generate demand for each
6    %pair of DCs
7    %tp = demand for packages for each pair of DCs
8    %For random number
9    dim  = size(tp,1);
10    %set the mean for each DC combination
11    mrndn = (ones(dim,dim)./tp)*24;
```

11. [loadpackages.m] is a MATLAB function which simulation loading of packages on a truck.

```
1    function [ev] =
loadpackages(currdcid,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,varargin)
2    %
3    %[trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] =
loadpackages(currdcid,ev,trk,pdc,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum)
4    %ev = object for event calander
5    %currdcid = current DC id
6    %tnow = current simulation time
7    %lut = loading/unlaoding time
8    %ve = not is use
9    %routes = cellarray habing shortest distance between each pair of DCs
10   %W = link flow
11   %waitMean = runnign average of the waiting time
12   %waitVar = running variance of waiting time
13   %waitNum = number of observations to calculate current mean waiting time
14   %A = matrix having distance between two directly conected DCs
15   %varargin = list of parameters
16   %   varargin{1} = truck id
17   %   varargin{2} = list of pack id to be loaded on the truck
18   global trkinfo;
19   global trk;
20   global pdc;
21   global allcosts;
22
23   oldbidamt = 0;
24   T = A/60;
25   pkgleave = [];
26   pval = 0;
27
28   %Get the current DC
29   cd = pdc(currdcid);
30   pksid = get(cd,'data');
31
32   leave= 1;
33   ntrkid = varargin{1};
34   ndcid = 0;
35   if length(varargin) == 2
36   pks = [];
37
38   %List of packages to be loaded on the truck
39   packset = varargin{2};
40
41
42   if size(packset,1) > 0
43   %Check if all the packages in the list are at the DC
44   packs = packset(:,1);
45   if size(pksid,1) > 0
```

```
46  pks = pksid(:,1);
47  end
48  c = ismember(pks,packs);
49  pksindav = find(c);
50
51  %If all the apckaegs in the list are not at the DC, do not laod the
52  %packages
53  if length(pksindav) ~= length(packs)
54  leave = 0;
55  pkgleave = [];
56  pksidtemp = pksid;
57
58  %If yes, load the packages
59  else
60  pkgleave = pksid(pksindav,:);
61  ndcid = routes{pkgleave(1,3),pkgleave(1,2)}(pkgleave(1,4)+1);
62  pkgindnotleave = setdiff([1:length(pks)],pksindav);
63  pksidtemp = pksid(pkgindnotleave,:);
64  [cd,oldbidamt] = updateTruck(ntrkid,cd);
65  end
66  %If list is empty, change the configuration of truck and DC directly
67  else
68  trknextdcs = get(trk(ntrkid),'dccont');
69  ndcid = trknextdcs(2);
70  pksidtemp = pksid;
71  leave = 1;
72
73  %change the configuration of truck
74  [cd,oldbidamt] = updateTruck(ntrkid,cd);
75  end
76  cdcs = get(cd,'connecteddcs');
77  nd = cdcs;
78  ndc = ndcid;
79  nd(nd ~= ndc) = 0;
80  ndin = find(nd);
81  am = ndin;
82
83
84  if size(pkgleave,1) > 0
85  %If number of packages leaving a DC is greate than zero
86
87  %set the statistics for the package to be shipped
88  pkgleave(:,7) = pkgleave(:,7) + (tnow - pkgleave(:,8));
89  totbidamtall = sum(pkgleave(:,9));
90  trkinfo(ntrkid,3) = trkinfo(ntrkid,3) - oldbidamt;
91  trkinfo(ntrkid,9) = trkinfo(ntrkid,9) - (totbidamtall - pval);
92
93  %set the variables for truck
94  trk(ntrkid) = set(trk(ntrkid),size(pkgleave,1),'currnumpack');
95
```

```
96   %set the variables for DC
97   cd = set(cd,size(pksidtemp,1),'currnumpack');
98   cd = set(cd,size(pkgleave,1),am,0,'numpacksuppbay');
99   cd = set(cd,sum(pkgleave(:,9)),am,0,'packsuppbay');
100   cd = set(cd,pksidtemp,'setdata');
101
102   pkage = pkgleave;
103
104   %increase the number of intermediate DC by 1
105   pkage(:,4) = pkage(:,4) + 1;
106
107   %get the packages whose final desination is not next DC
108   pktemp = pkage(:,[4 5]);
109   pktemp(pktemp(:,1) == pktemp(:,2)) = 0;
110   mnr = find(pktemp(:,1));
111   mnrpkg = pkage(mnr,:);
112
113   %get the packages whose final desination is next DC
114   pktemp = pkage(:,[4 5]);
115   pktemp(pktemp(:,1) ~= pktemp(:,2)) = 0;
116   mr = find(pktemp(:,1));
117   mrpkg = pkage(mr,:);
118   trk(ntrkid) = set(trk(ntrkid),mrpkg,'pkgnext');
119   trk(ntrkid) = set(trk(ntrkid),mnrpkg,'pkgnotnext');
120   end
121
122
123   if leave == 1
124   if size(pkgleave,1) > 0
125   pkgtrk = get(cd,'packtrk');
126   [x,y] = setdiff(pkgtrk(:,1),pkgleave(:,1));
127   if length(find(y)) > 0
128   cd = set(cd,pkgtrk(y,:),'setpacktrk');
129   else
130   cd = set(cd,[],'setpacktrk');
131   end
132   else
133   end
134
135   tcs = dijk(A,currdcid,ndcid)*allcosts(1);
136   trkinfo(ntrkid,10) = trkinfo(ntrkid,10) - tcs;
137   trk(ntrkid) = set(trk(ntrkid),ndcid,'ndcn');
138
139   %remove the truck from the queue
140   cd= removetruck(cd,ntrkid);
141   incomingt = get(cd,'incomingtrucks');
142   incoming = incomingt(:,1);
143   incoming(incoming ~= varargin{1}) = 0;
144   in = find(incoming);
145   ind = setdiff([1:length(incoming)],in);
```

```
146  cd = set(cd,incomingt(ind,:),'setincomingtrucks');
147
148  %schedule the departure of the truck
149  ev = cancelEvent(ev,[ntrkid 0]);
150  ev = putnextevent(ev,[tnow+lut/60,ntrkid,0]);
151
152  transtime = T(currdcid,ndcid);
153  trkinfo(ntrkid,1) = tnow+transtime+2*lut;
154  trkinfo(ntrkid,2) = ndcid;
155
156  %Add to new dc
157  trkdc = trkinfo(:,2);
158  trkdc(trkdc ~= ndcid) = 0;
159  trkdcin = find(trkdc);
160  trktnum = trkinfo(trkdcin,1);
161  [trktnumt,indt] = sort(trktnum);
162  trkdcinfi = trkdcin(indt);
163  trkinfo(trkdcinfi,6) = [1:length(trkdcinfi)]';
164
165  %Remove from old dc
166  trkdc = trkinfo(:,2);
167  trkdc(trkdc ~= currdcid) = 0;
168  trkdcin = find(trkdc);
169  trktnum = trkinfo(trkdcin,1);
170  [trktnumt,indt] = sort(trktnum);
171  trkdcinfi = trkdcin(indt);
172  trkinfo(trkdcinfi,6) = [1:length(trkdcinfi)]';
173
174  trk(ntrkid) = set(trk(ntrkid),1,'loading');
175
176  if currdcid == trkinfo(ntrkid,7)
177  trkinfo(ntrkid,7) = 0;
178  trkinfo(ntrkid,8) = 0;
179  end
180  end
181
182  pdc(currdcid) = cd;
```

12. [nextarrival.m] is a MATLAB function which simulates generation of demand.

```
1    function [packarrevent,packid,ev] =
nextarrival(packarrevent,packid,mrndn,ev,IJD2,tnow,m,routes)
2    %
3    %
4    %[packarrevent,packid,ev] =
nextarrival(packarrevent,pid,packid,mrndn,ev,IJD2,tnow,m,routes)
5    %packarrevent = an array having all the informaiton about packages
6    %packid = id of next package
7    %mrndn = an array having mean of distribution to generate demand for each pair of DCs
8    %ev= an object to maintain event calendar for simulation
9    %IJD2 = IJD matrix for a PLN
10   %tnow = current time of the simulation
11   %m = row id for the current package
12   %routes = cell array having shortest route between each pair of DCs
13
14
15   %get the source Dc and destination DC
16   custdc = packarrevent(m,2);
17   suppdc = packarrevent(m,3);
18
19   if m == size(packarrevent,1)
20   packarrevent = packarrevent(1:m-1,:);
21   elseif m == 1
22   packarrevent = packarrevent(2:end,:);
23   else
24   packarrevent = [packarrevent(1:m-1,:);packarrevent(m+1:end,:)];
25
26
27   %get the next package id
28   packid = packid + 1;
29
30   %get the time for next demand arival
31   narrt = genrandnum(mrndn(custdc,suppdc),'exp');
32   nextarrivaltime = tnow+narrt;
33
34   %Schedule the arrival of next demand
35   ev = putnextevent(ev,[tnow+narrt packid 3]);
36
37   %put the information regarding the package scheduled to be generated in
38   %a seperate matrix
39   nodc = length(routes{suppdc,custdc});
40   packarrevent = [packarrevent;packid custdc suppdc 0 nodc 0 0 0];
```

13. [pln.m] is a MATLAB function which provides information about PLN configuration.

```
1    function [tp,TSI,W] =  pln(pop,xy,IJD,TC,ts,p,lu,td,lf)
2    %
3    %[NT,AT,tp] = pln(pop,xy,IJD,TC,ts,p,lu,td)
4    %pop    -   Population matrix(36X1) for  each DC
5    %xy     -   Location matrix(36X2) for each DC
6    %IJD    -   Matrix containing the tail node, the head node and the distances between the
tail and the head nodes
7    %TC     -   n-element Truck Capacity Vector
8    %ts     -   Speed of the truck
9    %p      -   m-element Proximity factor vector, contains different possible proximity
factors
10   %lu     -   l-element loading/unloading time vector, contains different possible loading
unloading time in minutes
11   %td     -   Total demand (number of packages)
12   %lf     -   Load factor
13
14   %Sort the matrices supplied
15   TC = sort(TC);
16   p = sort(p);
17   lu = sort(lu);
18
19   %Generate the demand at all DCs from the population around it.
20   dem = (pop*100)/sum(pop);
21
22   % Generate the demand from DC j at DC i
23   w = dem*dem';
24   dim = size(w,1);
25   prox1{1} =  useProximity(p(1),w,xy);
26   tp = prox1{1}*td;
27
28   %Get IJT matrix for all loading and unloading time
29   for j = 1:length(lu)
30   IJT = IJD;
31   IJT(:,3) = IJD(:,3)/ts;
32   lut{j} = IJT;
33   end
34
35   %Get the demand matrix according to the proximity factor and the loading/unloading
time
36   for l = 1:length(p)
37   disp('Generating data...')
38   for q = 1:length(lu)
39   %Get all the routes
40   A = full(list2adj(lut{q}));
41   [D,P]=dijk(A);
42   rte=pred2path(P);
43
44   %Get the demand for each arc in the network
```

```
45  w3 = zeros(dim,dim);
46  for i=1:dim
47  for j=1:dim
48
49  %Considers all the possible paths from P(pred2path)
50  b=rte{i,j};
51  if i == j
52  b = [b b];
53  end
54
55  %Adds the demand of the route to the demand of the DC on the route
56  for k=1:length(b)-1
57  w3(b(k),b(k+1)) = w3(b(k),b(k+1)) + prox1{l}(i,j);
58  end
59  end
60  end
61  prox{l,q} = w3;
62  end
63  end
64  W = w3;
65  m = 0;
66  for y = 1:length(p)
67  for z = 1:length(TC)
68  m = m + 1;
69  n = 0;
70  for s = 1:length(lu)
71  n = n + 1;
72
73  %get the number of trucks
74  NT = genAvgTransTime(TC(z),prox{y,s},sum(sum(tp)),IJT,lf);
75
76
77  %get the truck distribution for different DCs based on
78  %the demand
79  TSI = wtselect(dem/100,1,ceil(sum(sum(NT))));
80  end
81  end
82  end
```

14. [simulation.m] is a MATLAB function which formats the input before starting the simulation.

```
1     function [AT,WT] =
simulation(nopack,trkcap,trksp,lu,pf,pt,wup,runlen,s,IJD2,xy,pop,lf,pick,tp,TSI,W,trkamt,cos
t,w,B)
2     %
3     %[AT,WT] =
simulation(nopack,trkcap,trksp,lu,pf,pt,wup,runlen,s,IJD2,xy,pop,lf,pick,tp,TSI,W,trkamt,cos
t,w,i,B)
4     %AT = Average transportation time from simulation
5     %WT = Average waiting time from simulation
6     %nopack = total packages in the system per day
7     %trkcap = truck capacity
8     %trksp = truck speed
9     %lu = loadin/unloading time
10    %pf = proximity factor
11    %wup = simulation warm-up period
12    %runlen = simulation run length
13    %s = seed in simulation
14    %IJD2 = IJD matrix for simulation
15    %xy = XY coordinate of each DC
16    %pop = population around each DC
17    %lf = load factor
18    %pick = way a queue of pakcages is picked
19    %tp = total package deand between each pair of DCs
20    %TSI = vector having DC id assigned to each truck
21    %W = link flow
22    %trkamt = not used
23    %cost = a vector with different costs
24    %   cost(1) = transportation cost
25    %   cost(2) = Labor cost
26    %w = packages between each pair of DCs
27    %B = package values
28
29    %Truck speed
30    TK.trkspeed = trksp;
31    %truck capacity
32    TK.mxtrkcap = trkcap;
33    %load factor
34    TK.loadfact = lf;
35    %Truck distribution at different DCs
36    TK.tsi = TSI;
37    %Not used now
38    TK.trkamt = trkamt;
39
40    %total package in the system per day
41    PK.total = nopack;
42    %total packages between diffferent DCs
43    PK.tp = tp;
```

```
%packages for each route
PK.w = W;
PK.ar = w;

DC.XY = xy;
DC.Pop = pop;

%loading/unloading time at DC
DC.lut = lu;
%Proximity factor
DC.prxf = pf;
%Process time for demand
DC.ppt= pt;

%Simulation warmup period
SIM.wp = wup;
%Simulation rum length
SIM.rl = runlen;
%Seed in simulation
SIM.seed = s;

[AT,WT] = startsimulation(DC,TK,PK,SIM,IJD2,pick,cost,B);
```

15. [start.m] is a MATLAB function where the input is provided for the simulation.

```
function start
load plnex36;
load Beta36;
IJD2 = IJD;

%truck capacity
trkcap = [10];
%trkamt - not used now
trkamt = 0;
 %number of packages per day in the system
 nopack = 20000*ones(1,15);
 %Number of trucks in the system
 nt = 72;
 %truck speed
 trkspeed = [60];
 %loading/unloading time
 lut = [0];
 %proximity factor
 pf = [0];
 %processing time in DC
 pt = 0;
 %simulation warm-up period
 warmupperiod = 0;
 %simulation run length
 runlength = 50;
 %load factor
 lf = [0.8];

 %Not used now
 qpick = [6];                %1 for maximum value or number queue, 0 for weighted
                             %selection, 2 for the scheduled version
                        %3 for maximum number of packages in a
                        %queue, 4 for maximum value and top n
                        %packages in terms of values, 5 for
                        %selecting top n bids from all queues using
                        %cost same as their values, 6 for Vickrey's
                        %Auction
 %Not used now
 %Fuel Costs $/mile
 fc = 2;
 %Labor Cost $/hr
 lc = 720000000000;
 cost = [fc lc];
 for i = 1:length(nopack)
 for qp = 1:length(qpick)
 for p = 1:length(seed)
 for j = 1:length(trkcap)
 for k = 1:length(trkspeed)
 for m = 1:length(lut)
```

```
49    for n = 1:length(pf)
50    for q = 1:length(lf)
51    %For PLN
52    [tp,TSI,W,w] =
pln(DC.Pop,DC.XY,IJD2,trkcap(j),trkspeed(k),pf(n),lut(m),nopack(i),lf(q),nt);
53    %For simulation
54    [ATT,WT] =
simulation(nopack(i),trkcap(j),trkspeed(k),lut(m),pf(n),pt,warmupperiod,runlength,seed(p),IJ
D2,DC.XY,DC.Pop,lf(q),qpick(qp),tp,TSI,W,trkamt,cost,w,BV{1,i});
55    end
56    end
57    end
58    end
59    end
60    end
61    end
62    end
```

16. [startsimulation.m] is a MATLAB function which is the main simulation function.

```
1    function  [ATT,avgwt] = startsimulation(DC,TK,PK,SIM,IJD2,pick,cost,B)
2    %
3    %[ATT,avgwt] = startsimulation(DC,TK,PK,SIM,IJD2,pick)
4    %ATT = Average transportation time from simulation
5    %avgwt = Average waiting time from simulation
6    %DC = Information about DCs in a PLN
7    %TK = Information about trucks in a PLN
8    %PK = Information about packages in aa PLN
9    %SIM= Information about simulation
10   %IJD2 = IJD matrix for a PLN
11   %pick = way a queue of packages is picked - (not used in this simulation)
12   %cost = transportation cost and labor cost
13   %B = vector having package values
14
15   %Setting the seed for the simulation run
16   rand('state',9501293);
17   permseedval = SIM.seed;
18   A = full(list2adj(IJD2));
19
20   %initialize global variables
21   [ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK);
22
23   %Initialize random numbers
24   [mrndn] = intializerandomnumbers(tp);
25
26   global trkinfo;
27   global trk;
28   global pdc;
29   global tnow;
30   global routes;
31   global allcosts;
32   global trkbidtime;
33   global trkbidamt
34   global trkbidndc;
35   global trkbidloadtime;
36   global totpackcost;
37   global packtrk;
38   totpackcost = 0;
39   packtrk = [];
40   allcosts = cost;
41
42   %initialize simulation
43   [ev,trk,pdc,packarrevent,allpackids,packid,T,routes,trkinfo] =
initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W,PK,B);
44
45   trkbidtime = zeros(length(trk),size(T,2));
46   trkbidamt = zeros(length(trk),size(T,2));
47   trkbidndc = zeros(length(trk),size(T,2));
48   trkbidloadtime = zeros(length(trk),size(T,2));
```

```
49
50  %For statistics collection and plotting graphs
51  si = size(tp);
52  intArrMean = zeros(si);
53  intArrVar = zeros(si);
54  intArrNum = zeros(si);
55  intArrOld = zeros(si);
56  loadMean = zeros(si);
57  loadVar = zeros(si);
58  loadNum = zeros(si);
59  waitMean = zeros(si);
60  waitVar = zeros(si);
61  waitNum = zeros(si);
62
63  endsim = 0;
64  count = 0;
65  totpack = 0;
66  totval = 0;
67  totval2 = 0;
68  told = 0;
69  global totpackval;
70
71  totpackval = 0;
72  packsto = {};
73
74  avgtrkld = [0 0];
75  avgtt = [0 0];
76  avgdcld = [0 0];
77  avgwtp = [0 0];
78  avgwt = 0;
79  td = 0;
80  ATT = 0;
81  totearn = 0;
82  intcount = 0;
83  totval = 0;
84  ve = [];
85  onedc = 0;
86  onedcc = 0;
87
88  % 1 - truck arrive
89  % 0 - truck leaving
90  % 3 - package arriving
91  % 4 - package leaving
92
93  disp('Simulation started...');
94  time = 1;
95  while(1)
96  [ev,rs] = getnextevent(ev,1);
97
98  %If there are no events in the event calendar
```

```
99    if size(rs,1) == 0
100   disp('No more events...');
101   ATT = td/rd;
102   break;
103   end
104
105   %If there are events in the event calendar, continue
106   r = rs(1,:);
107   tnow = r(1);
108
109   %truck arriving at a DC
110   if r(3) == 1
111   ev = putnextevent(ev,[tnow+lut/60,r(2),6]);
112   elseif r(3) == 6
113   %get the truck id
114   tr = r(2);
115
116   %Get the DC id at which truck arrives
117   if get(trk(tr),'cdcn') == 0
118
119   %for trucks coming from other DC
120   currdcid = get(trk(tr),'ndcn');
121   trk(tr) = set(trk(tr),currdcid,'cdcn');
122   trk(tr) = set(trk(tr),0,'ndcn');
123   else
124
125   %for the trucks coming straight to the DC. This happens once, at the beginning of the
simulation.
126   currdcid = get(trk(tr),'cdcn');
127   trk(tr) = set(trk(tr),0,'ndcn');
128   end
129
130   truckreached = tr;
131
132   %unload the packages
133   unloadpackages(tr,currdcid,tnow,routes,A);
134
135   %set the truck in the queue at a DC
136   pdc(currdcid) = set(pdc(currdcid),tr,'truckid');
137   truckids = get(pdc(currdcid),'truckid');
138
139   for i = 1:length(truckids)
140   if get(trk(truckids(i)),'loading') == 0
141   dccont = (get(trk(truckids(i)),'dccont'));
142   if length(dccont) > 0
143   dccont = dccont(1);
144   if currdcid == dccont
145   packsto = get(trk(truckids(i)),'pkgcont');
146   packstoload = packsto{1,1};
```

```
147   [ev] =
loadpackages(currdcid,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,truckids(i),pac
kstoload);
148   elseif dccont ~= 0
149   nextdc = routes{currdcid,dccont}(2)
150   [ev] =
loadpackages(currdcid,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,truckids(i),ne
xtdc,1);
151   end
152   end
153   end
154   end
155   if abs(tnow - 0.1) < 0.00001
156   %Bidding Process for the package and change in the configuration of the system.
157   nstep{1,1} = currdcid;
158   nstep{1,2} = [1:length(get(pdc(currdcid),'connecteddcs'))];
159   nstep{1,3} = tnow;
160   nsteo{1,4} = [];
161   furdc = [];
162   infopkg = get(pdc(currdcid),'info');
163
164   %Bidding at the DC for load formation
165   [furdc,tr,out,nstepn] = bidding(nstep{1,1},routes,T,tnow,1,nstep);
166   furdc = unique(furdc);
167   outt = {};
168   if size(tr,1) > 0
169   trucksel = trk(tr);
170   trkp = get(trucksel,'pkgcont');
171   if get(trucksel,'loading') == 0
172   cudc = get(trucksel,'cdcn');
173   firstdccont = get(trucksel,'dccont');
174   if cudc == firstdccont(1)
175   packsto = get(trucksel,'pkgcont');
176   packstoload = packsto{1};
177
178   %load the packages on the truck which are at DC where it has reservation
179   [ev] =
loadpackages(cudc,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,tr,packstoload);
180   end
181   end
182   end
183   end
184   %truck leaving a DC
185   elseif r(3) == 0
186   nstep = {};
187
188   %get the truck id
189   tr = r(2);
190   trk(tr) = set(trk(tr),0,'loading');
191
```

```
192  %set the final property of the truck before its departure and
193  %put the next event of the arrival of the truck, at the next DC, in the event calendar
194  cdc = get(trk(tr),'cdcn');
195  ndc = get(trk(tr),'ndcn');
196  transtime = T(cdc,ndc);
197  pkgtrans = get(trk(tr),'pkgnotnext');
198  pdc(ndc) = set(pdc(ndc),[tr tnow+transtime],'incomingtrucks');
199  cdcs = get(pdc(ndc),'connecteddcs');
200  if size(pkgtrans,1) > 0
201  nstpk = cell(1,length(cdcs));
202
203  %change the properties and bid amount for the packages whose final
204  %destination is not next DC
205  for k=1:size(pkgtrans,1)
206  routes{pkgtrans(k,3),pkgtrans(k,2)};
207  ndcid = routes{pkgtrans(k,3),pkgtrans(k,2)}(pkgtrans(k,4)+1);
208  nd = cdcs;
209  pkgtrans(k,9) = pkgtrans(k,10)*A(ndc,ndcid)/dijk(A,pkgtrans(k,3),pkgtrans(k,2));
210  nd(nd ~= ndcid) = 0;
211  ndin = find(nd);
212  pdc(ndc) = set(pdc(ndc),[pkgtrans(k,1) pkgtrans(k,9) tnow+transtime],ndin,'infot');
213  pdc(ndc) = set(pdc(ndc),[pkgtrans(k,1) 0],'packtrk');
214  nstpk{1,ndin} = [nstpk{1,ndin};pkgtrans(k,1)];
215  end
216  pkgtrans(:,11) = tnow + transtime;
217  trk(tr) = set(trk(tr),pkgtrans,'pkgnotnext');
218  end
219  nstep = {};
220  nstep{1,1} = ndc;
221  nstep{1,2} = 1:length(cdcs);
222  nstep{1,3} = [];
223  nstep{1,4} = [];
224
225  trk(tr) = set(trk(tr),0,'cdcn');
226  trk(tr) = set(trk(tr),cdc,'ldcn');
227
228  truckleaving = tr;
229
230  ev = putnextevent(ev,[tnow+transtime,tr,1]);
231
232  %Bidding Process and change in the configuration of the system becaue of new load
formation.
233  out1 = {};
234  bidocc = 0;
235
236  while size(nstep,1) > 0
237  nstepn = nstep(1,:);
238  if size(nstep,1) > 1
239  nstep = nstep([2:end],:);
240  else
```

```
241  nstep = {};
242  end
243  [furdct,trid,out,nstept] = bidding(nstepn{1,1},routes,T,tnow,1,nstepn);
244  nstep = [nstep;nstept];
245  if size(trid,1) > 0
246  furdc = [furdc furdct];
247  trucksel = trk(trid);
248  trkp = get(trucksel,'pkgcont');
249  if get(trucksel,'loading') == 0
250  cudc = get(trucksel,'cdcn');
251  firstdccont = get(trucksel,'dccont');
252  if cudc == firstdccont(1)
253  packsto = get(trucksel,'pkgcont');
254  packstoload = packsto{1,1};
255  [ev] =
loadpackages(cudc,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,trid,packstoload);
256  end
257  end
258  end
259  end
260  %package demand arriving
261  elseif r(3) == 3
262  ev = putnextevent(ev,[tnow,r(2),4]);
263  %processed package arriving
264  elseif r(3) == 4
265  nstep = {};
266  %get the package id
267  pid = r(2);
268  allpack = packarrevent(:,1);
269  allpack(allpack ~= pid) = 0;
270  m = find(allpack);
271  %get the source DC and destination DC
272  currdc = packarrevent(m,3);
273  destdc = packarrevent(m,2);
274  infopkg = get(pdc(currdc),'infot');
275  if currdc == destdc
276  %to satisfy local demand
277  rd = rd + 1;
278  tdlat = 0;
279  wtlat = 0;
280  tdold = td;
281  wtold = avgwt;
282  valold = totpackval;
283  currpackval = genrandnum(0,'unif',1);
284  td = (tdold*valold + tdlat*currpackval)/(valold+currpackval);
285  avgwt = (wtold*valold + wtlat*currpackval)/(valold+currpackval);
286  totpackval = totpackval + currpackval;
287  else
288  packarrevent(m,4) = 1;
289  packarrevent(m,6) = r(1);
```

```
290   packarrevent(m,7) = 0;
291   packarrevent(m,8) = r(1);
292   genran = genrandnum(0,'unif',1);
293   packarrevent(m,10) = genran*dijk(A,currdc,destdc);
294   %Find the next DC for the package
295   ndc = routes{currdc,destdc}(2);
296   dcs = get(pdc(currdc),'connecteddcs');
297   dcst = dcs;
298   dcst(dcst ~= ndc) = 0;
299   j = find(dcst);
300   packarrevent(m,9) =
packarrevent(m,10)*A(currdc,ndc)/dijk(A,packarrevent(m,3),packarrevent(m,2));
301
302   nstep{1,1} = currdc;
303   nstep{1,2} = j;
304   nstep{1,3} = tnow;
305   nstep{1,4} = pid;
306
307   %Update DC information
308   pdc(currdc) = set(pdc(currdc),packarrevent(m,:),'data');
309   pdc(currdc) = set(pdc(currdc),get(pdc(currdc),'currnumpack')+1,'currnumpack');
310   pdc(currdc) = set(pdc(currdc),1,j,1,'numpacksuppbay');
311   pdc(currdc) = set(pdc(currdc),packarrevent(m,9),j,1,'packsuppbay');
312   pdc(currdc) = set(pdc(currdc),[packarrevent(m,1) packarrevent(m,9) tnow],j,'infot');
313   packagearrived = pid;
314   pdc(currdc) = set(pdc(currdc),[pid 0],'packtrk');
315
316   %Bidding Process for the package and change in the configuration of the system.
317   furdc = [];
318   [furdc,tr,out,nstepn] = bidding(nstep{1,1},routes,T,tnow,1,nstep);
319   furdc = unique(furdc);
320   outt = {};
321   if size(tr,1) > 0
322   trucksel = trk(tr);
323   trkp = get(trucksel,'pkgcont');
324   if get(trucksel,'loading') == 0
325   cudc = get(trucksel,'cdcn');
326   firstdccont = get(trucksel,'dccont');
327   if cudc == firstdccont(1)
328   packsto = get(trucksel,'pkgcont');
329   packstoload = packsto{1};
330   [ev] =
loadpackages(cudc,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,tr,packstoload);
331   end
332   end
333   end
334   while size(nstepn,1) > 0
335   nstep = nstepn(1,:);
336   if size(nstepn,1) > 1
337   nstepn = nstepn([2:end],:);
```

```
338   else
339   nstepn = {};
340   end
341   [furdct,trid,out,nstept] = bidding(nstep{1,1},routes,T,tnow,1,nstep);
342   nstepn = [nstepn;nstept];
343   if size(trid,1) > 0
344   trucksel = trk(trid);
345   trkp = get(trucksel,'pkgcont');
346   if get(trucksel,'loading') == 0
347   cudc = get(trucksel,'cdcn');
348   firstdccont = get(trucksel,'dccont');
349   if cudc == firstdccont(1)
350   packsto = get(trucksel,'pkgcont');
351   packstoload = packsto{1};
352   [ev] =
loadpackages(cudc,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,trid,packstoload);
353   end
354   end
355   end
356   end
357   end
358   [packarrevent,packid,ev] =
nextarrival(packarrevent,pid,packid,mrndn{currdc,destdc},ev,IJD2,tnow,m,routes);
359   else r(3) == 5    %draw the graph
360   avgload = 0;
361   avgdcload = 0;
362   avgwttime = 0;
363   tfl = 0;
364   for j = 1:length(trk)
365   avgload = avgload + get(trk(j),'currnumpack');
366   end
367   avgload = avgload/length(trk);
368   for km = 1:length(pdc)
369   avgdcload = avgdcload + get(pdc(km),'currnumpack');
370   end
371   avgdcload = avgdcload/length(pdc);
372   tnow = r(1)
373   avgwttime = avgwt
374   ATT = td
375   ATL = avgload
376   ADL = avgdcload
377   ev = putnextevent(ev,[r(1)+1 5 5]);
378   for i = 1:length(pdc)
379   pdata = get(pdc(i),'data');
380   if size(pdata,1) > 0
381   for j = 1:size(pdata,1)
382   tdlat = tnow;
383   wtlat = tnow;
384   tdold = td;
385   wtold = avgwt;
```

```
386  valold = totpackval;
387  currpackval = pdata(j,10);
388  td = (tdold*valold + tdlat*currpackval)/(valold+currpackval);
389  avgwt = (wtold*valold + wtlat*currpackval)/(valold+currpackval);
390  totpackval = valold+currpackval;
391  end
392  end
393  end
394  for i = 1:length(trk)
395  pdata = get(trk(i),'data');
396  if size(pdata,1) > 0
397  for j = 1:size(pdata,1)
398  tdlat = tnow;
399  wtlat = tnow;
400  tdold = td;
401  wtold = avgwt;
402  valold = totpackval;
403  currpackval = pdata(j,10);
404  td = (tdold*valold + tdlat*currpackval)/(valold+currpackval);
405  avgwt = (wtold*valold + wtlat*currpackval)/(valold+currpackval);
406  totpackval = valold+currpackval;
407  end
408  end
409  end
410  fid3 = fopen('DataCollection','w+')
411  fprintf(fid3,'%6.4f \t',td);
412  fprintf(fid3,'%6.4f \t',avgwt);
413  fprintf(fid3,'%6.4f \t',ATL);
414  fprintf(fid3,'%6.4f \n',ADL);
415  fclose(fid3);
416  endsim = 1;
417  break;
418  %Warm up period ends
419  elseif r(3) == -2
420  td = 0;
421  rd = 0;
422  avgwt = 0;
423  end
424  if endsim == 1
425  break;
426  end
427  end
```

17. [unloadpackages.m] is a MATLAB function which simulates unloading operation.

```
1    function unloadpackages(tr,currdcid,tnow,routes,A)
2    %
3    %unloadpackages(tr,currdcid,tnow,routes,A)
4    %tr = truck id
5    %currdcid = current DC id
6    %tnow = current simulation time
7    %routes = cell array having routes for each pair of DCs
8    %A = matrix having distance between two directly connected DCs
9    global trk;
10   global pdc;
11   global totpackval;
12   global totpackcost;
13   global trkinfo;
14
15   mnrpkg = get(trk(tr),'pkgnotnext');
16   mrpkg = get(trk(tr),'pkgnext');
17   trk(tr) = set(trk(tr),[],'pkgnotnext');
18   trk(tr) = set(trk(tr),[],'pkgnext');
19   if size(mrpkg,1) > 0
20   for k = 1:size(mrpkg,1)
21   tdlat = tnow - mrpkg(k,6);
22   wtlat = mrpkg(k,7);
23   dis = dijk(A,mrpkg(k,3),mrpkg(k,2));
24   bidamt = mrpkg(k,10)/dis;
25   tdold = td;
26   wtold = avgwt;
27   valold = totpackval;
28   currpackval = mrpkg(k,10);
29   totpackcost = totpackcost + tdlat*currpackval;
30   td = (tdold*valold + tdlat*currpackval)/(valold+currpackval);
31   avgwt = (wtold*valold + wtlat*currpackval)/(valold+currpackval);
32   totpackval = valold+currpackval;
33   end
34   end
35   %If this is the intermediate DC
36   if size(mnrpkg,1) > 0
37   mnrpkg(:,8) = tnow;
38   mnrpkg(:,11) = 0;
39   %set the pakcages in the DC
40   pdc(currdcid) = set(pdc(currdcid),mnrpkg,'data');
41   %set the variables for DC
42   pdc(currdcid) =
set(pdc(currdcid),get(pdc(currdcid),'currnumpack')+size(mnrpkg,1),'currnumpack');
43   dcs = get(pdc(currdcid),'connecteddcs');
44   %increae the number of packages in the respective bays where packages are stored to be
transported to their next DCs
45   for i = 1:size(mnrpkg,1)
46   ndc = routes{mnrpkg(i,3),mnrpkg(i,2)}(mnrpkg(i,4)+1);
47   dcst = dcs;
```

```
48  dcst(dcst ~= ndc) = 0;
49  j = find(dcst);
50  pdc(currdcid) = set(pdc(currdcid),1,j,1,'numpacksuppbay');
51  pdc(currdcid) = set(pdc(currdcid),mnrpkg(i,9),j,1,'packsuppbay');
52  end
53  end
54  %set the number of packages in the truck to 0
55  trk(tr) = set(trk(tr),0,'currnumpack');
```

18. [updateTrucks.m] is a MATLAB function which updates trucks and DC information after packages are loaded on a truck.

```
1    function unloadpackages(tr,currdcid,tnow,routes,A)
2    %
3    %unloadpackages(tr,currdcid,tnow,routes,A)
4    %tr = truck id
5    %currdcid = current DC id
6    %tnow = current simulation time
7    %routes = cell array having routes for each pair of DCs
8    %A = matrix having distance between two directly connected DCs
9    global trk;
10   global pdc;
11   global totpackval;
12   global totpackcost;
13   global trkinfo;
14
15   mnrpkg = get(trk(tr),'pkgnotnext');
16   mrpkg = get(trk(tr),'pkgnext');
17   trk(tr) = set(trk(tr),[],'pkgnotnext');
18   trk(tr) = set(trk(tr),[],'pkgnext');
19   if size(mrpkg,1) > 0
20   for k = 1:size(mrpkg,1)
21   tdlat = tnow - mrpkg(k,6);
22   wtlat = mrpkg(k,7);
23   dis = dijk(A,mrpkg(k,3),mrpkg(k,2));
24   bidamt = mrpkg(k,10)/dis;
25   tdold = td;
26   wtold = avgwt;
27   valold = totpackval;
28   currpackval = mrpkg(k,10);
29   totpackcost = totpackcost + tdlat*currpackval;
30   td = (tdold*valold + tdlat*currpackval)/(valold+currpackval);
31   avgwt = (wtold*valold + wtlat*currpackval)/(valold+currpackval);
32   totpackval = valold+currpackval;
33   end
34   end
35   %If this is the intermediate DC
36   if size(mnrpkg,1) > 0
37   mnrpkg(:,8) = tnow;
38   mnrpkg(:,11) = 0;
39   %set the pakcages in the DC
40   pdc(currdcid) = set(pdc(currdcid),mnrpkg,'data');
41   %set the variables for DC
42   pdc(currdcid) =
set(pdc(currdcid),get(pdc(currdcid),'currnumpack')+size(mnrpkg,1),'currnumpack');
43   dcs = get(pdc(currdcid),'connecteddcs');
44   %increae the number of packages in the respective bays where packages are stored to be
transported to their next DCs
45   for i = 1:size(mnrpkg,1)
46   ndc = routes{mnrpkg(i,3),mnrpkg(i,2)}(mnrpkg(i,4)+1);
```

```
47  dcst = dcs;
48  dcst(dcst ~= ndc) = 0;
49  j = find(dcst);
50  pdc(currdcid) = set(pdc(currdcid),1,j,1,'numpacksuppbay');
51  pdc(currdcid) = set(pdc(currdcid),mnrpkg(i,9),j,1,'packsuppbay');
52  end
53  end
54  %set the number of packages in the truck to 0
55  trk(tr) = set(trk(tr),0,'currnumpack');
```

19. [useProximity.m] is a MATLAB function which helps in demand generation based on proximity factor.

```
function w = useProximity(p,w,xy)
%
%w = useProximity(p,w,xy)
%p  -   Proximity Factor
%w  -   Initial weight matrix for demand.
%xy -   DC location matrix.
w2 = w;
dis = dists(xy,xy,'mi');
[y,in] = sort(dis,2);
 for j = 1:size(w,1)
 for k = 1:size(w,1)
 w(j,k) = w2(j,in(j,k));
 end
 end

 %Compute the denominator
 td = 0;
 m = size(w,1);
 for k = 1:m
 td = td + power(1-(p/m),k-1);
 end
 td = td/m;

 %Get the numerator
 for i = 1:m
 for j= 1:m
 w1(i,j) = w(i,j)*power(1-(p/m),j-1);
 end
 end
 w1 = w1/td;
 wp = w1/sum(sum(w1));

 for j = 1:size(w,1)
 for k = 1:size(w,1)
 w(j,in(j,k)) = wp(j,k);
 end
 end
```

20. [changeConfig.m] is a MATLAB function which calculates the amount offered to a truck due to changes in load formations at various DCs and if truck accepts the bids at various DCs, changes the configuration of trucks and DCs.

```
1   function [furdc,tr,done,nstepn,finind,cvimp] = changeConfig(outfin,tnow,T)
2   %
3   %[furdc,tr,done,nstepn,finind,cvimp] = changeConfig(outfin,tnow,T)
4   %furdc = list of DCs where load formation has changed
5   %tr = truck id that has accepted the new load at various DCs
6   %done = "1" if truck has accepted new reservations at DCs
7   %       "0" if the truck rejects new reservations at DCs
8   %nstepn = information about the DCs where load formation has changed
9   %finind = not used now
10  %cvimp = if value is "-", the iteration stops
11  %outfin = information about the various laod formations at various DCs
12  %tnow = current simulation time
13  %T = matrix having travelling time between directly connected DCs.
14
15  global trkinfo;
16  global trk;
17  global pdc;
18  global allcosts;
19  global trkbidtime;
20  global trkbidamt;
21  global trkbidndc;
22  global trkbidloadtime;
23  global packtrk;
24
25
26  cvimp = 0;
27  done = 0;
28  lastrow = outfin(size(outfin,1),:);
29  currdcid = lastrow{3};
30  currdcindex = lastrow{2};
31  currdctime = lastrow{10};
32  trt = lastrow{1};
33  timefirst = -1;
34  firsttimemon = 0;
35
36  if length(lastrow{11}) ~= 0
37  timefirst = lastrow{11};
38  firsttimemon = trkinfo(trt,3) - trkinfo(trt,10);
39  end
40  if abs(tnow - trkbidtime(trt,currdcid)) < 0.000001 & trkinfo(trt,2) ==
trkbidndc(trt,currdcid)
41  timefirst = trkbidloadtime(trt,currdcid);
42  firsttimemon = trkbidamt(trt,currdcid);
43  end
44
45  final = {};
```

```
finalc = [];
finaltvm = [];
finaltm = [];
nstepn = {};
trkinfoti = [];
trkpkj = {};
trkdcj = [];
currdcid = [];
outres = {};
firstrows = {};
lastrow = {};
oldlastrow4col = {};
if size(outfin,1) > 0
if size(outfin,1) > 1
firstrows = outfin(1:size(outfin,1)-1,:);
end
lastrow = outfin(size(outfin,1),:);
currdcid = lastrow{3};
lastrow2col = lastrow{1,2};
lastrow8col = lastrow{1,8};
lastrow4col = lastrow{1,4};
lastrow10col = lastrow{1,10};
lastrow11col = lastrow{1,11};
outt = cell(1,10);
%Going through various load formations at various DCs.
for fi = 1:size(lastrow2col,1)
outt{1} = lastrow{1};
outt{2} = lastrow2col(fi,:);
outt{3} = lastrow{3};
outt{4} = lastrow4col{fi,1};
%Checking if load information is empty
if cmpcell(outt{4},oldlastrow4col) == 0
continue;
end
oldlastrow4col = outt{4};
outt{8} = lastrow8col(fi,:);
outt{9} = lastrow{9};
outt{10} = lastrow10col(fi);
outt{11} = lastrow11col;
out = [firstrows;outt];
outres = out;
finind = 0;
furdc = [];
if size(out,1) > 0
ct = trk(out{size(out,1),1});
tr = out{size(out,1),1};
trkinfoti = trkinfo(tr,:);
trkpkj = get(trk(tr),'pkgcont');
trkdcj = get(trk(tr),'dccont');
pkst1234 = out{size(out,1),4};
```

```
96   tcount = 0;
97   trdcold= get(trk(tr),'dccont');
98   outtwo = out(size(out,1),:);
99   outone = {};
100  if size(out,1) > 1
101  outone = out([1:size(out,1)-1],:);
102  end
103  outtwopack = outtwo{1,4};
104  outtwobid = outtwo{1,8};
105  [a,b] = sort(outtwobid);
106  out = outone;
107  si = size(outone,1) + 1;
108  totnumbid = length(b);
109  finalout = {};
110  finalbidout = [];
111  finalbidouttwo = [];
112  finalbidoutcost = [];
113  finaloutcost = [];
114  ldcbid  = 0;
115
116  %For different load formations and their bids at a DC
117  for s = size(b,2):-1:1
118  out{si,1} = outtwo{1,1};
119  out{si,2} = outtwo{1,2}(b(s));
120  out{si,3} = outtwo{1,3};
121  out{si,4} = outtwopack{b(s)};
122  out{si,5} = [];
123  out{si,6} = [];
124  out{si,7} = [];
125  out{si,8} = a(s);
126  out{si,9} = [];
127  out{si,10} = outtwo{1,10};
128  if length(outtwo{1,9}) > 0
129  outtwoone = outtwo{1,9};
130  if out{si,2} == outtwoone(2)
131  out{si,9} = outtwoone(1);
132  end
133  end
134  totlatbid = 0;
135  flg = 0;
136  totbidamt = 0;
137  totoldbid = 0;
138  pkgcontnew = {};
139  pkgcontnewtwo = {};
140  pkgcontold = {};
141  dccontnew = [];
142  dccontnewtemp = [];
143  dccontold = [];
144  indextondcnew = [];
145  indextondcold = [];
```

```
146  inbidamt = [];
147  inbidamtold = [];
148  inbidamtnew = [];
149  index = 0;
150  index2 = 0;
151  count = 0;
152  dc = [];
153  dcpkg = [];
154  oldpacks = [];
155  dcindex = [];
156  pkgcont = get(ct,'pkgcont');
157  dccont = get(ct,'dccont');
158  indextondc = get(ct,'indextonextdc');
159  inbidamt = get(ct,'bids');
160  mult = 0;
161  for i=1:size(out,1)
162
163  %If truck does not has reservations with some of the intermediate DCs
164  if out{i,1} ~= 0
165
166  %If new bay number is not same as old bay number for the truck
167  if length(out{i,9}) == 0
168  %Form arrays and cell arrays with information about new load
169  count = count + 1;
170  pkgcontnew{1,count} = out{i,4};
171  %list of new DCs
172  dccontnew = [dccontnew out{i,3}];
173  %list of new bay numbers
174  indextondcnew = [indextondcnew out{i,2}];
175  totbidamt = totbidamt + out{i,8};
176  %list of amount recieved by truck from various DCs
177  inbidamtnew = [inbidamtnew out{i,8}];
178  %If new bay number is same as old bay number for the truck
179  else
180  %Form arrays and cell arrays with information about new load
181  inbidamtnew = inbidamt;
182  mult = 1;
183  dc = out{i,3};
184  dcindex = out{i,2};
185  dccontin = out{i,9};
186  packs = get(trk(tr),'pkgcont');
187  pack = [];
188  packbidid = [];
189  pack = packs{1,dccontin};
190  packid = [];
191  if size(pack,1) > 0
192  packid = pack(:,1);
193  end
194  packbid = out{i,4};
195  if size(packbid,1) > 0
```

```
196  packbidid = packbid(:,1);
197  end
198  [diff,in] = setdiff(packbidid,packid);
199  dcinfo = get(pdc(dccont(dccontin)),'info');
200  dcinfot = dcinfo{1,out{i,2}};
201  dcinfo = get(pdc(dccont(dccontin)),'info');
202  if length(in) > 0
203  newload = packbid(in,:);
204  dcinfot = dcinfo{1,out{i,2}};
205  if size(dcinfot,1) > 0
206  if size(newload,1) > 0
207  dcinfott = dcinfot(:,1);
208  [mem,intt] = setdiff(dcinfott,newload(:,1));
209  dcinfottt = [];
210  if length(intt) > 0
211  dcinfottt = dcinfot(intt,:);
212  end
213  dcpkg = dcinfottt;
214  end
215  end
216  else
217  dcpkg = dcinfo{1,out{i,2}};
218  end
219  [diff,in] = setdiff(packid,packbidid);
220  totoldbid = 0;
221  oldpacks = [];
222  if length(in) > 0
223  dcpkg = [dcpkg;pack(in,:)];
224  totoldbid = totoldbid + sum(pack(in,2));
225  oldpacks= [oldpacks;pack(in,:)];
226  end
227  [mem,in] = ismember(packid,packbidid);
228  infd =  find(in);
229  in = in(infd);
230  if length(in) > 0
231  totoldbid = totoldbid + sum(packbid(in,2));
232  end
233  packs{1,dccontin} = out{i,4};
234  pkgcontnew = packs;
235  transswer = get(trk(tr),'pkgcont');
236  pks = out{i,4};
237  if size(pks,1) > 0
238  totlatbid = trkinfo(tr,9) + sum(pks(:,2)) - totoldbid;
239  end
240  dcpkgc{1,1} = dcpkg;
241  if currdcid == dccont(end)
242  trkinff = trkinfo(tr,1);
243  if tnow - trkinff > 0.00001
244  trkinff = tnow;
245  end
```

```
trdc = trkinff + dijk(T,trkinfo(tr,2),currdcid);
if currdcid == dccont(end)
trdc = trkinfo(tr,8);
end
if  ~abs(trdc - outtwo{1,10}) < 0.00001
flg = 1;
totoldbid = 0;
inbidamtnew = [];
[val,index2] = ismember(out{i,3},dccont);

%list of new DCs
dccontnew = dccont(1:index2);

%list of old DCs
dccontold = dccont(index2:end);

%List of new bay numbers
indextondcnew = indextondc(1:index2);

%list of old bay number
indextondcold = indextondc(index2:end);

%inbidamtnew - amount rruck recieves from different DCs
if index2 > 1
inbidamtnew = inbidamt(1:index2-1);
end
if size(pks,1) > 0
inbidamtnew = [inbidamtnew sum(pks(:,2))];
else
inbidamtnew = [inbidamtnew 0];
end
for j = 1:size(pkgcont,2)
if j < index2
pkgcontnewtwo{1,j} = pkgcont{1,j};
elseif j == index2
pkgcontnewtwo{1,j} = out{i,4};
pkgcontold{1,1} = oldpacks;
else
pkgcontold{1,j-index2+1} = pkgcont{1,j};
pk = pkgcontold{1,j-index2+1};
if size(pkgcontold{1,j-index2+1},1) > 0
totoldbid = totoldbid + sum(pk(:,2));
end
end
end
pkgcontnew = pkgcontnewtwo;
totlatbid = totlatbid - totoldbid;
end
end
%If truck already has reservations with some of the intermediate DCs
```

```
296  else
297  dcc = out{i,3};
298  [val,index] = ismember(dcc,dccont);
299  end
300  end
301  flag = 0;
302
303  %replace packages for the overlapping DC between new and old bids
304  if mult ~= 1
305  if length(dccont) > 0
306  if dccontnew(1) == dccont(end)
307  if indextondcnew(1) == indextondc(end)
308  pack = [];
309  packbidid = [];
310  pack = pkgcont{1,end};
311  packid = [];
312  if size(pack,1) > 0
313  packid = pack(:,1);
314  end
315  packbid = pkgcontnew{1,1};
316  if size(packbid,1) > 0
317  packbidid = packbid(:,1);
318  end
319  oldpacks = [];
320  [diff,in] = setdiff(packid,packbidid);
321  if length(in) > 0
322  oldpacks = pack(in,:);
323  end
324  flag = 1;
325  end
326  end
327  end
328  end
329  if mult == 0
330  if index == 0
331  pkgcontold = pkgcont;
332  if flag == 1
333  pkgcontold{1,end} = oldpacks;
334  end
335  dccontold = dccont;
336  indextondcold = indextondc;
337  totoldbid = 0;
338  if size(pkgcontold,2) > 0
339  for i = 1:size(pkgcontold,2)
340  if flag == 1 & i == size(pkgcontold,2)
341  pk = pkgcont{1,end};
342  else
343  pk = pkgcontold{1,i};
344  end
345  if size(pk,1) > 0
```

```
totoldbid = totoldbid + sum(pk(:,2));
end
end
end
else
for i = 1:size(pkgcont,2)
if i <= index
pkgcontnewtwo{1,i} = pkgcont{1,i};
else
pkgcontold{1,i-index} = pkgcont{1,i};
if  i == size(pkgcont,2)
if flag == 1
pkgcontold{1,end} = oldpacks;
end
end
if flag == 1 & i == size(pkgcont,2)
pk= pkgcont{1,end};
else
pk = pkgcontold{1,i-index};
end
if size(pk,1) > 0
totoldbid = totoldbid + sum(pk(:,2));
end
end
end
if index ~= length(dccont)
dccontold = dccont(index+1:end);
indextondcold = indextondc(index+1:end);
end
dccontnew = [dccont(1:index) dccontnew];
indextondcnew = [indextondc(1:index) indextondcnew];
inbidamtnew = [inbidamt(1:index) inbidamtnew];
for i = 1:size(pkgcontnew,2)
pkgcontnewtwo{1,index+i} = pkgcontnew{1,i};
end
pkgcontnew = pkgcontnewtwo;
end
dccontnew;
totlatbid = trkinfo(tr,9) + totbidamt - totoldbid;
indval = totnumbid-s+1;

%putting the entrie information in a cell array
finalout{indval,1} = dccontnew;
finalout{indval,2} = dccontold;
finalout{indval,3} = indextondcnew;
finalout{indval,4} = indextondcold;
finalout{indval,5} = pkgcontnew;
finalout{indval,6} = pkgcontold;
finalout{indval,7} = inbidamtnew;
finalout{indval,8} = [];
```

```
finalbidouttwo(indval,1) = totlatbid;
finalbidout(indval,1) = sum(inbidamtnew);
else
indval = totnumbid-s+1;
if flg == 0
%putting the entrie information in a cell array
finalout{indval,1} = dccont;
finalout{indval,2} = dcindex;
finalout{indval,3} = [];
finalout{indval,4} = dc;
finalout{indval,5} = pkgcontnew;
finalout{indval,6} = dcpkgc;
finalout{indval,7} = inbidamtnew;
finalout{indval,8} = [];
finalbidouttwo(indval,1) = totlatbid;
finalbidout(indval,1) = sum(inbidamtnew);
elseif flg == 1
%putting the entrie information in a cell array
finalout{indval,1} = dccontnew;
finalout{indval,2} = dccontold;
finalout{indval,3} = indextondcnew;
finalout{indval,4} = indextondcold;
finalout{indval,5} = pkgcontnew;
finalout{indval,6} = pkgcontold;
finalout{indval,7} = inbidamtnew;
finalout{indval,8} = index2;
finalbidouttwo(indval,1) = totlatbid;
finalbidout(indval,1) = sum(inbidamtnew);
end
end
tc = 0;

%calculate the trasnportation cost for the enitre stretch of journey
if length(finalout{indval,3}) > 0
indexndctc = finalout{indval,3};
dcconttc = finalout{indval,1};
ldc = dcconttc(end);
ldctcin = indexndctc(end);
conteddc = get(pdc(ldc),'connecteddcs');
lldc = conteddc(ldctcin);
tc = (dijk(T*60,dcconttc(1),ldc)+dijk(T*60,ldc,lldc))*allcosts(1);
else
tc = trkinfo(tr,10);
end
finalbidoutcost(indval,1) = finalbidout(indval,1) - tc;
finaloutcost(indval,1) = tc;
end

%get the series of DCs and bay numbers which gives maximum amount of money
%to the truck
```

```
446 [bidmax,bidind] = max(finalbidout);
447 cv = finalbidoutcost(bidind);
448 cvimp = cv;
449 cvo = trkinfo(tr,3) - trkinfo(tr,10);
450 if cvo < 0
451 cvo = 0;
452 end
453 cv1 = trkinfo(tr,9);
454 cv2 = finalbidouttwo(bidind);
455 ldct = finalout{bidind,1};
456 ldc = ldct(end);
457 if ismember(currdcid,finalout{bidind,1}) == 1
458 ti = outtwo{1,10} + dijk(T,outtwo{1,3},ldc);
459 else
460 ti = outtwo{1,10};
461 end
462 ldco = trkinfo(tr,7);
463 tio = trkinfo(tr,8);
464 if totlatbid < 0
465 totlatbid;
466 end
467 cvlast = 0;
468 tclast = 0;
469 pkgsindcs = finalout{bidind,5};
470 pkgsinlastdc = pkgsindcs{1,size(pkgsindcs,2)};
471 if size(pkgsinlastdc,1) > 0
472 pkgsbidsinlastdc = pkgsinlastdc(:,2);
473 cvlast = sum(pkgsbidsinlastdc,1);
474 if size(finalout{bidind,3}) > 0
475 indexndctc = finalout{bidind,3};
476 else
477 indexndctc = indextondc;
478 end
479 dcconttc = finalout{bidind,1};
480 ldc = dcconttc(end);
481 ldctcin = indexndctc(end);
482 conteddc = get(pdc(ldc),'connecteddcs');
483 lldc = conteddc(ldctcin);
484 tclast = dijk(T*60,ldc,lldc)*2;
485 end
486 if tclast == 0
487 tclast = 999999999;
488 end
489 if done ~= 2
490
491 %Check if net amount truck gets is more than the current amount it gets
492 if cvlast - tclast > 0.00001
493 if length(dccont) > 0
494 dccontnews = finalout{bidind,1};
495 if dccont(end) == dccontnews(end)
```

```
496  if currdcid ~= dccontnews(end)
497  if cv2 - cv1 > 0.00001
498  done= 2;
499  end
500  elseif currdcid == dccontnews(end)
501  if ~abs(trkinfo(tr,8) - outt{10}) < 0.00001
502  cvlast = inbidamtnew(end);
503  if cv > 0.00001
504  if cvlast - tclast > 0.00001
505  done = 2;
506  end
507  end
508  elseif abs(trkinfo(tr,8) - outt{10}) < 0.00001
509  indexndctc = indextondc;
510  dcconttc = dccont;
511  ldc = dcconttc(end);
512  ldctcin = indexndctc(end);
513  conteddc = get(pdc(ldc),'connecteddcs');
514  lldc  = conteddc(ldctcin);
515  tclastold = dijk(T*60,ldc,lldc)*2;
516  cvlastold = inbidamt(end);
517  cvlast = inbidamtnew(end);
518  if cv > 0.00001
519  if ((cvlast-tclast) - (cvlastold - tclastold)) > 0.00001
520  done = 2;
521  elseif abs((cvlast-tclast) - (cvlastold - tclastold)) < 0.00001
522  if cv2 - cv1 > 0.00001
523  done = 2;
524  end
525  end
526  end
527  end
528  end
529  else
530  if length(finalout{bidind,8}) == 0
531  if cv - cvo > 0.00001
532  done = 2;
533  elseif abs(cv-cvo) < 0.00001
534  if cv2 - cv1 > 0.00001
535  done = 2;
536  end
537  end
538  else
539  index2 = finalout{bidind,8};
540  pkg2new = finalout{bidind,5};
541  pkg2oldin = pkgcont{1,index2};
542  pkg2newin = pkg2new{1,end};
543  if cmparr(pkg2oldin,pkg2newin) == 1
544  if cv - cvo > 0.00001
545  done = 2;
```

```
546  elseif abs(cv-cvo) < 0.00001
547  if cv2 - cv1 > 0.00001
548  done = 2;
549  end
550  end
551  end
552  end
553  end
554  else
555  if cvlast > 0
556  if length(finalout{bidind,8}) == 0
557  if cv - cvo > 0.00001
558  done = 2;
559  elseif abs(cv-cvo) < 0.00001
560  if cv2 - cv1 > 0.00001
561  done = 2;
562  end
563  end
564  else
565  index2 = finalout{bidind,8};
566  pkg2new = finalout{bidind,5};
567  pkg2oldin = pkgcont{1,index2};
568  pkg2newin = pkg2new{1,end};
569  if cmparr(pkg2oldin,pkg2newin) == 1
570  if cv - cvo > 0.00001
571  done = 2;
572  elseif abs(cv-cvo) < 0.00001
573  if cv2 - cv1 > 0.00001
574  done = 2;
575  end
576  end
577  end
578  end
579  end
580  end
581  end
582  end
583
584  %If truck gets more money than its current amount
585  if done == 2
586  if timefirst == -1
587  %change the configuration of the truck
588  timefirst = outt{10};
589  firsttimemon = cv;
590  final = finalout;
591  finalc = finaloutcost(bidind);
592  finaltm = finalbidout(bidind);
593  finaltvm = finalbidouttwo(bidind);
594  end
595  if length(finalout{bidind,3}) == 0
```

```
donem = 0;
if cv - cvo > 0.00001
donem = 1;
elseif abs(cv-cvo) < 0.00001
if cv2 - cv1 > 0.00001
donem = 1;
end
end
if donem == 1
if abs(outt{10} - timefirst) < 0.00001
final = finalout;
finalc = finaloutcost(bidind);
finaltm = finalbidout(bidind);
finaltvm = finalbidouttwo(bidind);
else
cvnew = cv - (outt{10} - timefirst)*allcosts(2);
if cvnew - firsttimemon > 0.00001
timefirst = outt{10};
firsttimemon = cv;
final = finalout;
finalc = finaloutcost(bidind);
finaltm = finalbidout(bidind);
finaltvm = finalbidouttwo(bidind);
end
end
end
else
cvnew = cv - (outt{10} - timefirst)*allcosts(2);
if cvnew - firsttimemon > 0.00001
final = finalout;
finalc = finaloutcost(bidind);
finaltm = finalbidout(bidind);
finaltvm = finalbidouttwo(bidind);
timefirst = outt{10};
firsttimemon = cv;
end
end
end
end
end
finalout = {};
finaloutcost = [];
finalbidout = [];
finalbidouttwo = [];
bidind = 1;
finalout = final;
finaloutcost = finalc;
finalbidout = finaltm;
finalbidouttwo = finaltvm;
if size(final,1) > 0
```

```
646 if length(final{bidind,1}) == 1
647 trkinfo(tr,10) = finaloutcost(bidind,1);
648 dccontbid  = finalout{bidind,1};
649 dcconttr = get(trk(tr),'dccont');
650 if length(dcconttr) > 0
651 if dccontbid(end) ~= dcconttr(end)
652 trkinfo(tr,7) = outt{3};
653 trkinfo(tr,8) = timefirst;
654 elseif currdcid == dccontbid(end)
655 trkinfo(tr,7) = outt{3};
656 trkinfo(tr,8) = timefirst;
657 end
658 else
659 trkinfo(tr,7) = outt{3};
660 trkinfo(tr,8) = timefirst;
661 end
662 done = 1;
663 if length(finalout{bidind,3}) > 0
664 dccontnew = finalout{bidind,1};
665 dccontold = finalout{bidind,2};
666 indextondcnew = finalout{bidind,3};
667 indextondcold = finalout{bidind,4};
668 pkgcontnew = finalout{bidind,5};
669 pkgcontold = finalout{bidind,6};
670 inbidamtnew = finalout{bidind,7};
671 otr = 0;
672 otr2 = [];
673 for i = 1:length(dccontnew)
674 dcinfo = get(pdc(dccontnew(i)),'info');
675 dcinfot = dcinfo{1,indextondcnew(i)};
676 pkg = pkgcontnew{1,i};
677 get(pdc(dccontnew(i)),'info');
678 if size(pkg,1) > 0
679 pkgtr = get(pdc(dccontnew(i)),'packtrk');
680 pkgtrt = pkgtr(:,1);
681 [x,y] = ismember(pkg(:,1),pkgtrt);
682 otr = pkgtr(y,2);
683 otr = unique(otr);
684 otr2 = [otr2;otr];
685 pkgtr(y,2) = tr;
686 pkgtrt = pkgtr(:,2);
687 pkgtrt(pkgtrt ~= tr) = 0;
688 a = find(pkgtrt);
689 pkgtrt = pkgtr(a,1);
690 [x,y] = setdiff(pkgtrt,pkg(:,1));
691 pkgtr(a(y),2) = 0;
692 pdc(dccontnew(i)) = set(pdc(dccontnew(i)),pkgtr,'setpacktrk');
693
694 %change the configuration of new DCs
695 if size(dcinfot,1) > 0
```

```
696  [diff,in] = setdiff(dcinfot(:,1),pkg(:,1));
697  if length(in) > 0
698  dcinfottwo = dcinfot(in,:);
699  pdc(dccontnew(i)) = set(pdc(dccontnew(i)),dcinfottwo,indextondcnew(i),'setinfo');
700  else
701  pdc(dccontnew(i)) = set(pdc(dccontnew(i)),[],indextondcnew(i),'setinfo');
702  end
703  else
704  pdc(dccontnew(i)) = set(pdc(dccontnew(i)),[],indextondcnew(i),'setinfo');
705  end
706  else
707  pkgtrk = get(pdc(dccontnew(i)),'packtrk');
708  if size(pkgtrk,1) ~= 0
709  pkgtrd = pkgtrk(:,1);
710  pkgtrt = pkgtrk(:,2);
711  pkgtrt(pkgtrt ~= tr) = 0;
712  a = find(pkgtrt);
713  if length(a) ~= 0
714  pkgtrk(a,2) = 0;
715  pdc(dccontnew(i)) = set(pdc(dccontnew(i)),pkgtrk,'setpacktrk');
716  end
717  end
718  end
719  end
720  otr = unique(otr2);
721
722  %Change the configuration of the truck
723  ct = set(ct,pkgcontnew,'pkgcont');
724  ct = set(ct,dccontnew,'dccont');
725  ct = set(ct,indextondcnew,'indextonextdc');
726  ct = set(ct,inbidamtnew,'bids');
727
728  %Change the configuration of the old dc
729  for j = 1:length(dccontold)
730  pdc(dccontold(j)) = set(pdc(dccontold(j)),pkgcontold{1,j},indextondcold(j),'info');
731  end
732  furdc = dccontold;
733  trk(out{size(out,1),1}) = ct;
734  else
735  trk(tr) = set(trk(tr),finalout{bidind,5},'pkgcont');
736  dcpkgf = finalout{bidind,6};
737  pdc(finalout{bidind,4}) =
set(pdc(finalout{bidind,4}),dcpkgf{1,1},finalout{bidind,2},'setinfo');
738  trk(out{size(out,1),1}) = set(trk(out{size(out,1),1}),finalout{bidind,7},'bids');
739  furdc = finalout{bidind,4};
740  [x,y] = ismember(currdcid,finalout{bidind,1});
741  pkgt = finalout{bidind,5};
742  pkg = pkgt{1,y};
743  pkgtr = get(pdc(currdcid),'packtrk');
744  pkgtrt = pkgtr(:,1);
```

```
745  [x,y] = ismember(pkg(:,1),pkgtrt);
746  otr = pkgtr(y,2);
747  otr = unique(otr);
748  pkgtr(y,2) = tr;
749  pkgtrt = pkgtr(:,2);
750  pkgtrt(pkgtrt ~= tr) = 0;
751  a = find(pkgtrt);
752  pkgtrt = pkgtr(a,1);
753  [x,y] = setdiff(pkgtrt,pkg(:,1));
754  if length(y) > 0
755  pkgtr(a(y),2) = 0;
756  end
757  pdc(currdcid) = set(pdc(currdcid),pkgtr,'setpacktrk');
758  end
759  otrl = length(find(otr));
760  if otrl ~= 0
761  for i = 1:length(otr)
762  if otr(i) ~= tr
763  if otr(i) ~= 0
764  trtk = otr(i);
765  dccon = get(trk(trtk),'dccont');
766  pkgcon = get(trk(trtk),'pkgcont');
767  indexton = get(trk(trtk),'indextonextdc');
768  inbidon = get(trk(trtk),'bids');
769  [x,y] = ismember(currdcid,dccon);
770  trk(trtk)= set(trk(trtk),[],'dccont');
771  trk(trtk) = set(trk(trtk),{},'pkgcont');
772  trk(trtk) = set(trk(trtk),[],'indextonextdc');
773  trk(trtk) = set(trk(trtk),[],'bids');
774  trkinfo(trtk,10) = 0;
775  trkinfo(trtk,8) = 0;
776  trkinfo(trtk,7) = 0;
777  y = 1;
778  for j = y:length(dccon)
779  pkg = pkgcon{1,j};
780  pkgtr = get(pdc(dccon(j)),'packtrk');
781  if size(pkgtr,1) > 0
782  pkgtrt = pkgtr(:,1);
783  if size(pkg,1) > 0
784  [x,z] = ismember(pkg(:,1),pkgtrt);
785  pkgtrn = pkgtr(z,2);
786  pkgtrn(pkgtrn ~= trtk) = 0;
787  b = find(pkgtrn);
788  if length(find(b)) ~= 0
789  pkgtr(z(b),2) = 0;
790  end
791  pdc(dccon(j)) = set(pdc(dccon(j)),pkgtr,'setpacktrk');
792  pkgsumr = sum(pkg(:,2));
793  pkgsumv =  inbidon(j);
794  trkinfo(trtk,3) = trkinfo(trtk,3) - pkgsumv;
```

```
795  trkinfo(trtk,9) = trkinfo(trtk,9) - pkgsumr;
796  end
797
798  %Arrange information about the DCs which have gone through new load
799  %formation
800  nstepn{j-y+1,1} = dccon(j);
801  nstepn{j-y+1,2} = indexton(j);
802  nstepn{j-y+1,3} = [];
803  nstepn{j-y+1,4} = [];
804  end
805  end
806  end
807  end
808  end
809  end
810  furdc = [furdc;currdcid];
811  if length(furdc) > 0
812  npkg = finalout{bidind,6};
813  if finalout{bidind,3} > 0
814  nindex = finalout{bidind,4};
815  else
816  nindex = finalout{bidind,2};
817  end
818  tn = size(nstepn,1);
819  for i = 1:length(furdc)
820  tn = tn + 1;
821  nstepn{tn,1} = furdc(i);
822  nstepn{tn,2} = [];
823  nstepn{tn,3} = [];
824  nstepn{tn,4} = [];
825  end
826  end
827  furdc = [currdcid furdc];
828  trkinfo(tr,3) = finalbidout(bidind);
829  trkinfo(tr,9) = finalbidouttwo(bidind);
830  trkbidtime(tr,currdcid) = tnow;
831  trkbidamt(tr,currdcid) = firsttimemon;
832  trkbidndc(tr,currdcid) = trkinfo(tr,2);
833  trkbidloadtime(tr,currdcid) = timefirst;
834  else
835  tr = [];
836  done = 0;
837  if length(dccont) > 0
838  for i = 1:length(dccont)
839  pkgc = pkgcont{1,i};
840  pkgdc = get(pdc(dccont(i)),'info');
841  pkgdcind = pkgdc{1,indextondc(i)};
842  if size(pkgdcind,1) > 0
843  if size(pkgc,1) > 0
844  pkgcid = pkgc(:,1);
```

```
845  pkgdcindid = pkgdcind(:,1);
846  [x,y] = setdiff(pkgdcindid,pkgcid);
847  pdc(dccont(i)) = set(pdc(dccont(i)),pkgdcind(y,:),indextondc(i),'setinfo');
848  end
849  end
850  end
851  end
852  end
853  else
854  tr = [];
855  done = 0;
856  if length(dccont) > 0
857  for i = 1:length(dccont)
858  pkgc = pkgcont{1,i};
859  pkgdc = get(pdc(dccont(i)),'info');
860  pkgdcind = pkgdc{1,indextondc(i)};
861  if size(pkgdcind,1) > 0
862  if size(pkgc,1) > 0
863  pkgcid = pkgc(:,1);
864  pkgdcindid = pkgdcind(:,1);
865  [x,y] = setdiff(pkgdcindid,pkgcid);
866  pdc(dccont(i)) = set(pdc(dccont(i)),pkgdcind(y,:),indextondc(i),'setinfo');
867  end
868  end
869  end
870  end
871  end
872  end
873  if done == 1
874  d = get(trk(tr),'dccont');
875  end
```