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

iv
LIST OF FIGURES

Figure 1.1: 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

Figure 1.2: Hypothetical Network of a PLN [1] ................................................................................. 3

Figure 1.3: A package transported from DC8 to DC1 .......................................................................... 4

Figure 1.4: A PLN with Distribution Centers, Pick-up Centers and Delivery Points .......... 5

Figure 1.5: Internet system showing Routers and local networks ................................................. 5

Figure 2.1: Flowchart showing various stages of package transportation in the simulation model ........................................................................................................ 13

Figure 2.2: Flowchart showing activities at the event of generation of demand .................. 15

Figure 2.3: Flowchart showing the activities at the event of package ready to be picked at the supplier DC ............................................................................................... 16

Figure 2.4: Flowchart showing the activities at the event of truck arrival at a DC ............ 17

Figure 2.5: Flowchart showing activities at the event of truck departure ......................... 18

Figure 2.6: 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

Figure 2.7: Exponential Equation fitted on the plot between ‘d’ and load factor ............... 23

Figure 3.1: Bid amount for package P1 from DC1 to DC2 and from DC2 to DC3 ........... 28

Figure 3.2: Load formation at DC3 ................................................................................................. 29

Figure 3.3: 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

Figure 3.4: Priority to accept and reject load at DC3 ................................................................. 33

Figure 3.5: Before accepting load at DC1, truck needs to accept load at DC5 and DC8 ... 34
Figure 3.6: 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

Figure 3.7: Different routes possible for T1 to reach DC1 ........................................35

Figure 3.8: Trucks end up getting less money because of shortest route constraint........36

Figure 3.9: Framework for the agent based co-ordination........................................39

Figure 3.10: Package arrival event in the simulation with protocol .........................40

Figure 3.12: Event of truck leaving a DC in the simulation with protocol..............41

Figure 3.11: Bidding Process......................................................................................42

Figure 3.13: Event of truck arriving at a DC in the simulation with protocol.........44

Figure 4.1: Packages at DC3 waiting to be transported to their respective next DCs ......45
LIST OF TABLES

Table 1.1: Costs for Transport of a Single Package from DC8 to DC1...........................4
Table 1.2: Comparison between a PLN and the Internet .................................................6
Table 2.1: Set of values for simulation runs ........................................................................19
Table 2.2: Average Waiting Time and Average Transportation Time (in hours) for demand of 50000 packages per day, truck capacity 20 and load factor 0.7 ............................19
Table 2.3: Average Waiting Time and Average Transportation Time (in hours) for demand of 70000 packages per day, truck capacity 100 and load factor 0.8 .........................20
Table 2.4: Shows relation between parameters ‘c’ and ‘d’ for different sets of demand, truck capacity and load factors ............................................................................................22
Table 2.5: R-square values for different equations on figure 2.7 ........................................22
Table 2.6: Verification of the Equation (4) by assigning demand randomly among DCs..25
Table 2.7: Verification of the Equation (4) by reassigning distances between DCs and also demand among DCs randomly ..................................................................................25
Table 2.8: Comparison of “Waiting Time from simulation” with Half Headway and Full Headway ....................................................................................................................26
Table 3.1: Allocation of load bid. ........................................................................................32
Table 3.2: Different possible routes for truck T2 to reach DC1 from DC5 and time taken to reach DC1 .................................................................................................................37
Table 3.3: Different possible routes for truck T2 to reach DC1 from DC5 and money it would receive from them .................................................................................................37
Table 3.4: Complete route, money offered by packages on each route, transportation cost and net amount received by truck T2 .................................................................37
Table 4.1: Packages in Q1 at DC3 and their time of arrival (in the increasing order) and amount offered ..................................................................................................................46
Table 4.2: Packages in Q2 at DC3 and their time of arrival (in the increasing order) and amount offered.................................................................46

Table 4.3: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time........................................48

Table 4.4: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time.................48

Table 4.5: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time........................................49

Table 4.6: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time.................49

Table 4.7: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time........................................50

Table 4.8: 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.

Figure 1.1: 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.
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.
Figure 1.3: A package transported from DC8 to DC1

Table 1.1: Costs for Transport of a Single Package from DC8 to DC1

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

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.

<table>
<thead>
<tr>
<th>PLN</th>
<th>Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution Centers</td>
<td>Routers</td>
</tr>
<tr>
<td>Physical Packages</td>
<td>Data Packets</td>
</tr>
<tr>
<td>Trucks</td>
<td>Wires/Cables</td>
</tr>
</tbody>
</table>

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 if 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 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 coordination 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\textsubscript{i} to DC\textsubscript{j} would imply an eight-hour average waiting time for every package at DC\textsubscript{i} transported through the link to DC\textsubscript{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.
Figure 2.1: 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.

   ![Flowchart](image)

   **Figure 2.2:** Flowchart showing activities at the event of generation of demand
Figure 2.3: 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

Figure 2.4: Flowchart showing the activities at the event of truck arrival at a DC
	heir 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.
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.
Table 2.1: Set of values for simulation runs

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

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

Table 2.2: Average Waiting Time and Average Transportation Time (in hours) for demand of 50000 packages per day, truck capacity 20 and load factor 0.7

<table>
<thead>
<tr>
<th>Seed for Random Number Generation</th>
<th>Average Waiting Time (overall)</th>
<th>Average Transportation Time (Overall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0.2424</td>
<td>6.931</td>
</tr>
<tr>
<td>6507440</td>
<td>0.2425</td>
<td>6.936</td>
</tr>
<tr>
<td>8661526</td>
<td>0.2423</td>
<td>6.9381</td>
</tr>
<tr>
<td>7976494</td>
<td>0.2424</td>
<td>6.9336</td>
</tr>
<tr>
<td>1568674</td>
<td>0.2426</td>
<td>6.9363</td>
</tr>
<tr>
<td>415885</td>
<td>0.2424</td>
<td>6.9381</td>
</tr>
<tr>
<td>4121782</td>
<td>0.2427</td>
<td>6.9355</td>
</tr>
</tbody>
</table>
Table 2.3: Average Waiting Time and Average Transportation Time (in hours) for demand of 70000 packages per day, truck capacity 100 and load factor 0.8

<table>
<thead>
<tr>
<th>Seed for Random Number Generation</th>
<th>Average Waiting Time (overall)</th>
<th>Average Transportation Time (Overall)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6721375</td>
<td>1.1276</td>
<td>7.8021</td>
</tr>
<tr>
<td>8317961</td>
<td>1.1398</td>
<td>7.4433</td>
</tr>
<tr>
<td>4610952</td>
<td>1.1325</td>
<td>7.5669</td>
</tr>
<tr>
<td>502689</td>
<td>1.1313</td>
<td>7.6591</td>
</tr>
<tr>
<td>7679504</td>
<td>1.1019</td>
<td>8.0812</td>
</tr>
<tr>
<td>4983114</td>
<td>1.1411</td>
<td>7.2729</td>
</tr>
</tbody>
</table>

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

\[ w = c \cdot x^{-a} \]  

(1)

where \( x \) - link flow

\( w \) - link average waiting time

\( c \) and \( a \) are parameters
Figure 2.6: 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:

$$c = \left( \frac{tc}{np} \right)^d$$  \hspace{1cm} (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 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.
Table 2.4: Shows relation between parameters ‘c’ and ‘d’ for different sets of demand, truck capacity and load factors

<table>
<thead>
<tr>
<th>Demand of Packages per Day</th>
<th>Truck Capacity</th>
<th>Load Factor</th>
<th>Coefficient (c) (from simulation)</th>
<th>Truck Cap/Demand of Packages (tc/np) = r</th>
<th>d = c/r</th>
</tr>
</thead>
<tbody>
<tr>
<td>30000</td>
<td>20</td>
<td>0.7</td>
<td>0.0038</td>
<td>0.0006666667</td>
<td>5.7</td>
</tr>
<tr>
<td>40000</td>
<td>20</td>
<td>0.7</td>
<td>0.0029</td>
<td>0.0005</td>
<td>5.8</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.5</td>
<td>0.0015</td>
<td>0.0004</td>
<td>3.75</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.5</td>
<td>0.0015</td>
<td>0.0004</td>
<td>3.75</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.55</td>
<td>0.0016</td>
<td>0.0004</td>
<td>4</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.6</td>
<td>0.0018</td>
<td>0.0004</td>
<td>4.5</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.65</td>
<td>0.0021</td>
<td>0.0004</td>
<td>5.25</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.67</td>
<td>0.0022</td>
<td>0.0004</td>
<td>5.5</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.7</td>
<td>0.0024</td>
<td>0.0004</td>
<td>6</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.7</td>
<td>0.0023</td>
<td>0.0004</td>
<td>5.75</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.7</td>
<td>0.0024</td>
<td>0.0004</td>
<td>6</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.7</td>
<td>0.0024</td>
<td>0.0004</td>
<td>6</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.72</td>
<td>0.0025</td>
<td>0.0004</td>
<td>6.25</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.75</td>
<td>0.0027</td>
<td>0.0004</td>
<td>6.75</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.77</td>
<td>0.0029</td>
<td>0.0004</td>
<td>7.25</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.8</td>
<td>0.0032</td>
<td>0.0004</td>
<td>8</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.8</td>
<td>0.0032</td>
<td>0.0004</td>
<td>8</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.82</td>
<td>0.0034</td>
<td>0.0004</td>
<td>8.5</td>
</tr>
<tr>
<td>50000</td>
<td>20</td>
<td>0.85</td>
<td>0.0038</td>
<td>0.0004</td>
<td>9.5</td>
</tr>
<tr>
<td>70000</td>
<td>20</td>
<td>0.5</td>
<td>0.0011</td>
<td>0.000285714</td>
<td>3.85</td>
</tr>
<tr>
<td>70000</td>
<td>20</td>
<td>0.7</td>
<td>0.0017</td>
<td>0.000285714</td>
<td>5.95</td>
</tr>
<tr>
<td>70000</td>
<td>20</td>
<td>0.8</td>
<td>0.0023</td>
<td>0.000285714</td>
<td>8.05</td>
</tr>
</tbody>
</table>

Step 4:

To find the relation between d and the load factor \( LF \), the values of d were 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.

Table 2.5: R-square values for different equations on figure 2.7

<table>
<thead>
<tr>
<th>Equation</th>
<th>R-square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0.9262</td>
</tr>
<tr>
<td>Quadratic</td>
<td>0.9956</td>
</tr>
<tr>
<td>Cubic</td>
<td>0.9961</td>
</tr>
<tr>
<td>Exponential</td>
<td>0.983</td>
</tr>
</tbody>
</table>
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.

![Graph showing 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:

\[ d = e^{2.6lf} \]  

(3)

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

\[ w = \left( \frac{te}{np} \right) \cdot e^{2.6lf} \cdot x \]  

(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:

\[
W = \left( \frac{tc}{np} \right) \cdot e^{2.66 + \lambda x - 0.96}
\]

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

\[
AW = \sum w_i x_i
\]

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

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

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

<table>
<thead>
<tr>
<th>Demand of packages per day</th>
<th>Truck Capacity</th>
<th>Load factor</th>
<th>Avg. Waiting Time (from Simulation)</th>
<th>Avg. Waiting Time (from new analytical model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50,000</td>
<td>100</td>
<td>0.8</td>
<td>1.5499</td>
<td>1.5649</td>
</tr>
<tr>
<td>70,000</td>
<td>100</td>
<td>0.7</td>
<td>1.0823</td>
<td>1.0824</td>
</tr>
</tbody>
</table>

**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.
Table 2.8: Comparison of “Waiting Time from simulation” with Half Headway and Full Headway

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

Figure 3.1: 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.
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.

![Diagram showing package protocol](image)

<table>
<thead>
<tr>
<th>Time</th>
<th>Load (in $)</th>
<th>Other Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30 10</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>30 10 7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>30 15 10 7</td>
<td>7 5</td>
</tr>
</tbody>
</table>

*Figure 3.3: 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.

<table>
<thead>
<tr>
<th>Package Event</th>
<th>Truck Response</th>
<th>Load Bid</th>
<th>Truck Portion</th>
<th>Allocated Portion</th>
<th>Load (Bid/Cost)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bid &amp; Join</td>
<td>Reject</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>7/7</td>
</tr>
<tr>
<td>Bid &amp; Join</td>
<td>Accept</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>7/7 3/3</td>
</tr>
<tr>
<td>Bid &amp; Join</td>
<td>–</td>
<td>15</td>
<td>10</td>
<td>5</td>
<td>7/4 5/5 3/1</td>
</tr>
<tr>
<td>Bid, Join &amp; Drop</td>
<td>–</td>
<td>18</td>
<td>10</td>
<td>8</td>
<td>7/-1 6/6 5/5</td>
</tr>
</tbody>
</table>

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.

![Diagram](image)

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).
Figure 3.5: Before accepting load at DC1, truck needs to accept load at DC5 and DC8

Figure 3.6: 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.

![Diagram](image)

Figure 3.7: 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.

![Figure 3.8: Trucks end up getting less money because of shortest route constraint](image-url)
Table 3.2: Different possible routes for truck T2 to reach DC1 from DC5 and time taken to reach DC1

<table>
<thead>
<tr>
<th>Route No.</th>
<th>Route</th>
<th>Time Taken</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DC5 – DC2 – DC1</td>
<td>0.8 hrs (0.3 hrs on DC5 to DC2 + 0.5 hrs on DC2 to DC1)</td>
</tr>
<tr>
<td>2</td>
<td>DC5 – DC3 – DC1</td>
<td>0.6 hrs (0.2 hrs on DC5 to DC3 + 0.4 hrs on DC3 to DC1)</td>
</tr>
</tbody>
</table>

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.

Table 3.3: Different possible routes for truck T2 to reach DC1 from DC5 and money it would receive from them

<table>
<thead>
<tr>
<th>Route No.</th>
<th>Route</th>
<th>Time Taken</th>
<th>Money Received by truck T2 (A - B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DC5 – DC2 – DC1</td>
<td>$80 ($30 on DC5 to DC2 + $50 on DC2 to DC1)</td>
<td>$56</td>
</tr>
<tr>
<td>2</td>
<td>DC5 – DC3 – DC1</td>
<td>$30 ($10 on DC5 to DC3 + $20 on DC3 to DC1)</td>
<td>$30</td>
</tr>
</tbody>
</table>

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.

Table 3.4: Complete route, money offered by packages on each route, transportation cost and net amount received by truck T2

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

![Diagram: 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.

![Diagram](image)

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.
Figure 3.12: 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.
Figure 3.13: 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.

![Diagram of packages at DC3 waiting to be transported to their respective next DCs](image-url)

Figure 4.1: 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.

Table 4.1: Packages in Q1 at DC3 and their time of arrival (in the increasing order) and amount offered

<table>
<thead>
<tr>
<th>Package</th>
<th>Time of Arrival</th>
<th>Amount Offered</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>P2</td>
<td>1.25</td>
<td>10</td>
</tr>
<tr>
<td>P3</td>
<td>1.5</td>
<td>30</td>
</tr>
<tr>
<td>P4</td>
<td>1.7</td>
<td>40</td>
</tr>
<tr>
<td>P5</td>
<td>2.0</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 4.2: Packages in Q2 at DC3 and their time of arrival (in the increasing order) and amount offered

<table>
<thead>
<tr>
<th>Package</th>
<th>Time of Arrival</th>
<th>Amount Offered</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6</td>
<td>1.2</td>
<td>50</td>
</tr>
<tr>
<td>P7</td>
<td>1.5</td>
<td>60</td>
</tr>
<tr>
<td>P8</td>
<td>1.6</td>
<td>30</td>
</tr>
<tr>
<td>P9</td>
<td>1.7</td>
<td>45</td>
</tr>
</tbody>
</table>

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.

Table 4.3: Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time

<table>
<thead>
<tr>
<th>Demand of Packages per day</th>
<th>Number of Trucks</th>
<th>Truck Capacity</th>
<th>Weighted Waiting Time (in hrs)</th>
<th>% change in Weighted Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Protocol</td>
<td>Without Protocol</td>
</tr>
<tr>
<td>10000</td>
<td>36</td>
<td>10</td>
<td>42.71</td>
<td>48.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>33.26</td>
<td>44.50</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>32.27</td>
<td>44.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>18.85</td>
<td>25.67</td>
</tr>
<tr>
<td>20000</td>
<td>36</td>
<td>10</td>
<td>47.00</td>
<td>49.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>42.55</td>
<td>48.49</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>42.07</td>
<td>48.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>31.81</td>
<td>43.91</td>
</tr>
</tbody>
</table>

Table 4.4: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time

<table>
<thead>
<tr>
<th>Demand of Packages per day</th>
<th>Number of Trucks</th>
<th>Truck Capacity</th>
<th>Weighted Transportation Time (in hrs)</th>
<th>% change in Weighted Transportation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Protocol</td>
<td>Without Protocol</td>
</tr>
<tr>
<td>10000</td>
<td>36</td>
<td>10</td>
<td>44.62</td>
<td>48.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>37.91</td>
<td>45.89</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>36.98</td>
<td>45.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>26.55</td>
<td>33.61</td>
</tr>
<tr>
<td>20000</td>
<td>36</td>
<td>10</td>
<td>47.74</td>
<td>49.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>44.52</td>
<td>48.78</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>44.17</td>
<td>48.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>36.66</td>
<td>45.54</td>
</tr>
</tbody>
</table>
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.

**Table 4.5:** Weighted Waiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time

<table>
<thead>
<tr>
<th>Demand of Packages per day</th>
<th>Number of Trucks</th>
<th>Truck Capacity</th>
<th>Weighted Waiting Time (in hrs)</th>
<th>% change in Weighted Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Protocol</td>
<td>Without Protocol</td>
</tr>
<tr>
<td>20000</td>
<td>36</td>
<td>10</td>
<td>49.05</td>
<td>49.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>46.99</td>
<td>48.49</td>
</tr>
<tr>
<td>20000</td>
<td>72</td>
<td>10</td>
<td>46.92</td>
<td>48.47</td>
</tr>
</tbody>
</table>

**Table 4.6:** Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time

<table>
<thead>
<tr>
<th>Demand of Packages per day</th>
<th>Number of Trucks</th>
<th>Truck Capacity</th>
<th>Weighted Transportation Time (in hrs)</th>
<th>% change in Weighted Transportation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Protocol</td>
<td>Without Protocol</td>
</tr>
<tr>
<td>20000</td>
<td>36</td>
<td>10</td>
<td>49.21</td>
<td>49.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>47.72</td>
<td>48.77</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>47.66</td>
<td>48.76</td>
</tr>
</tbody>
</table>

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.
Table 4.7: WeightedWaiting Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted waiting Time.

<table>
<thead>
<tr>
<th>Demand of Packages per day</th>
<th>Number of Trucks</th>
<th>Truck Capacity</th>
<th>Weighted Waiting Time (in hrs)</th>
<th>% change in Weighted Waiting Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Protocol</td>
<td>Without Protocol</td>
</tr>
<tr>
<td>20000</td>
<td>36</td>
<td>10</td>
<td>48.35</td>
<td>49.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>45.74</td>
<td>48.49</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>45.55</td>
<td>48.48</td>
</tr>
</tbody>
</table>

Table 4.8: Weighted Transportation Time for a PLN “Without Protocol” and “With Protocol” and percentage change in weighted transportation time.

<table>
<thead>
<tr>
<th>Demand of Packages per day</th>
<th>Number of Trucks</th>
<th>Truck Capacity</th>
<th>Weighted Transportation Time (in hrs)</th>
<th>% change in Weighted Transportation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Protocol</td>
<td>Without Protocol</td>
</tr>
<tr>
<td>20000</td>
<td>36</td>
<td>10</td>
<td>48.73</td>
<td>49.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>46.79</td>
<td>48.78</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>10</td>
<td>46.67</td>
<td>48.77</td>
</tr>
</tbody>
</table>

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


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.
  - `currcmpack`
    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.
  - `cdn`
    This is a scalar quantity. If the truck is at a DC, cdn 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

- decont
  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.

- indexonextdc
  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:
  - cdc
    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.
  - ddc
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.

- lden
  It gives the id of the last dc, the truck has been to.

- Maxcapacity
  It gives the maximum capacity of the truck.

- nden
  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.

- sden
  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.

- decont
  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.

- indexonextdc
  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.
  
  - **currmumpack**
    This is a scalar quantity and the value is the current number of packages in the DC at any point in time.
  
  - **connecteddc**
    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:
    
    - **connecteddc**
      This gives a vector containing ids of all the DCs connected to the DC. The first element is the id of the DC itself.
    
    - **currmumpack**
      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:
    
    - **currmumpack**
      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.
  
  - **trckid**
    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.
  
  - **packsuppby**
    This vector has same number of elements as the “connecteddc” 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.
  
  - **numpacksuppby**
This vector has same number of elements as the “connecteddc” 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.
  
  * packsuppby
    This gives the value of different queues. Each queue is for one DC directly connected to this DC.
  
  * numpacksuppby
    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.

. incomingtrucks
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.

```matlab
function c = createdc(n,IJD2,trk,W,pick)

seq = []; 
A = full(list2adj(IJD2));
disp('Generating DC....');

for i = 1:n
%get the list of connected DCs
conn = [i];
for j = 1:size(A,2)
if A(i,j) ~= 0
conn = [conn j];
end
end
if pick == 2
W1 = W(i,conn(2:end));
seq = fixseq(W1);
end
%set the number of packages in each bay as 0
suppbay = zeros(size(conn,1),size(conn,2));

%set the package ids as null
pid = [];

%set the truck ids as null
tid = [];

%for the DC object
  c(i) = plndc(i,inf,0,conn,suppbay,suppbay,pid,tid,[],seq);
end
```
2. [creatretrucks.m] is a MATLAB function which is used to create truck objects

```
function trk = createtrucks(TSI,TK)
%
% trk = createtrucks(TSI,TK)
% %trk = array of truck objects
% %TSI = vector containing DC ids assigned to trucks
% %TK = structure containing truck information.
% disp('Generating Trucks....');
for i = 1:length(TSI)
    trk(i) = truck(i,TSI(i),TSI(i),{},0,TK.mxtrkcap,0,TSI(i),TSI(i),TSI(i),0,0,0,0,0);
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,IJT,L,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 generates 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.

```matlab
1    function [ev,trk,pdc,packarrevent,allpackids,packid,T,routes] = 
2    initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W) 
3    %
4    %[ev,trk,pdc,packarrevent,allpackids,packid,T,routes] = 
5    initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W) 
6    %ev = object for event calendar 
7    %trk = array for truck object 
8    %pdc = array for plndc objects 
9    %packarrevent = array containing package information 
10   %T = matrix having time between each connecting DCs 
11   %routes = cell array for the shortest route between two DCs 
12   %tp = demand for package for each pari of DCs 
13   %IJD2 = IJD matrix for a PLN 
14   %mrndn = matrix having means to generate demand for each pair of DCs 
15   %TSI = vector having DC ids for different trucks at the start of the 
16   %simulation 
17   %SIM = values for the simulation 
18   %TK = information to create truck objects 
19   %pick = way a queue can be picked at a DC 
20   %W = link flow 
21   
22   A = full(list2adj(IJD2));
23   T = A/ts;
24   dim = size(A,1);
25   packid = 0;
26   
27   [d,pred] = dijk(list2adj(IJD2));
28   routes = pred2path(pred);
29   for i = 1:size(routes,1)
30      routes{i,i} = [routes{i,i} routes{i,i}];
31   end
32   
33   %create the trucks
34   trk = createtrucks(TSI,TK);
35   
36   %create event calendar
37   ev = eventcal;
38   
39   %create DCs
40   pdc = createdc(dim,IJD2,trk,W,pick);
41   
42   %For package taking in event calendar
43   packarrevent = [];
44   nt = length(trk);
45   tnow = 0;
46   disp('putting trucks arrival in the event calendar....');
47   for i = 1:nt
48      %1 for arrival of trucks at a DC
```

63
47  ev = putnextevent(ev,[0 i 1]);
48  end
49
50  %putting the formation of graph in the event calendar
51  ev = putnextevent(ev,[1 5 5]);
52  %putting the package arrival in the event calendar
53  allpack = [];
54  allpackids = [];
55  disp('putting packages in the event calendar....');
56  for i = 1:size(mrmdn,1)
57    for j = 1:size(mrmdn,2)
58      packid = packid + 1;
59      narrt = genrandnum(mrmdn(i,j),'exp');
60      ev = putnextevent(ev,[tnow+narrt,packid,3]);
61      nodc = length(routes(j,i));
62      packarrevent = [packarrevent,packid j 0 nodc 0 0 0];
63    end
64  end
65
66  ev = putnextevent(ev,[SIM.wp -2 -2]);
67  ev = putnextevent(ev,[SIM.r1 -1 -1]);
68  [ev,r] = getnextevent(ev,0);
6. [initializevariables.m] is a MATLAB function which initializes variables at the beginning of the simulation.

```matlab
function [ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK)

%tk = information to create truck objects
%dc = information to create dc objects
%pk = information to create package objects
%truck speed
%loading/unloading time
lut = DC.lut;

%processing time for packages
ppt = DC.ppt;

%break variable
out = 0;

%number of observations
rd = 0;

%total time
td = 0;

%package id
packid = 0;

%total package demand between different DCs
tp = PK.tp;

%truck distribution at different DCs
TSI = TK.tsi;

%percentage package distribution for each route
W = PK.w;
```
7. [initializerandomnumbers.m] is a MATLAB function which initializes random number generation.

```matlab
1    %This function initialize the variables used to generate random numbers.
2    function [mrndn] = initializerandomnumbers(tp)
3    %
4    %[mrndn] = initializerandomnumbers(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.

```matlab
function [trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] = loadpackages(currccd,cd,cd,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum)
% [trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] = loadpackages(currccd,cd,cd,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum)
% trk = array for truck objects
% pdc = array for pln dc objects
% ev = object for event calander
% W1 = link flow
% currccd = current DC id
% tnow = current simulation time
% lut = loading/unloading time
% pick = way to pick package queue
% ve = not is use
% routes = cellarray habing shortest distance between each pair of DCs
% W = link flow
% waitMean = runnign average of the waiting time
% waitVar = running variance of waiting time
% waitNum = number of observations to calculate current mean waiting time

% Get the current DC
cd = pdc(currccd);

% Get the next DC for which there is a queue with maximum value
ndcids = get(cd,'connecteddc');
v = [];

if pick == 1  
    % Queue with maximum value
    am = argmax(get(cd,'packsuppbay'));
    ndcid = ndcids(am);
    W1 = [];
    dcnext = get(cd,'numpacksuppbay');
    dcnext = dcnext,
elseif pick == 0
% With random pick
rndpick = 0;
ndcids1 = ndcids(2:end);
W1 = W(currccd,ndcids1);
while rndpick == 0
    ams = wtselect(W1,1,1);
    am = ams;
    am = am + 1;
    ndcid = ndcids(am);
    dcnext = get(cd,'numpacksuppbay');
    dcnext = dcnext,
end
```
47    rndpick = 1;
48    end
49    end
50
51    elseif pick == 2    %For scheduled truck arrival
52        seqpick = 0;
53        while seqpick == 0
54            currsq = get(cd,'currsqval');
55            cd = set(cd,'currsqval');
56            am = currsq + 1;
57            ndcid = ndcids(am);
58            dcnxts = get(cd,'numpacksuppby');
59            dcnxt = dcnxts(am);
60            if dcnxt == 0
61                seqpick = 1;
62                end
63                end
64                W1 = [];
65                end
66
67                %get the first truck from the DC queue
68                trkids = get(cd,'truckid');
69                ntrkid = trkids(1);
70
71                %set the number of packages that the truck will carry
72                if dcnxt > get(trk(ntrkid),'maxcapacity')
73                    numpack = get(trk(ntrkid),'maxcapacity');
74                else
75                    numpack = dcnxt;
76                end
77
78                %get all the packages in the DC
79                pkid = get(cd,'data');
80                pkidtemp = [];
81                j = [];
82                for i = 1:size(pkid,1)
83                    ndc = routes{pkid(i,3),pkid(i,2)}(pkid(i,4)+1);
84                    %select the package whose next destination is same as the selected next DC
85                    if ndc == ndcid
86                        j = [j;i];
87                        if size(j,1) == numpack
88                            break;
89                        end
90                    else
91                        pkidtemp = [pkidtemp,pkid(i,:)];
92                    end
93                end
94
95                pkidtemp = [pkidtemp,pkid((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        oldMean = waitMean(cdc,ndc);
109        waitMean(cdc,ndc) = (waitMean(cdc,ndc)*waitNum(cdc,ndc) +
110          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) +
113            waitNum(cdc,ndc)*power(waitMean(cdc,ndc)-oldMean,2) + power(waitMean(cdc,ndc)-
114              wait,2))/(waitNum(cdc,ndc)),0.5);
115        elseif waitNum(cdc,ndc) == 1
116          waitVar(cdc,ndc) = 0
117        end
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);
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.

```matlab
function [packarrevent,packid,ev] =
nextarrival(packarrevent,packid,mrndn,ev,IJD2,tnow,m,routes)

% [packarrevent,packid,ev] =
nextarrival(packarrevent,pid,packid,mrndn,ev,IJD2,tnow,m,routes)
% packarrevent = an array having all the informaiton about packages
% packid = id of next package
% mrndn = an array having mean of distribution to generate demand for each pair of DCs
% ev = an object to maintain event calendar for simulation
% IJD2 = IJD matrix for a PLN
% tnow = current time of the simulation
% m = row id for the current package
% routes = cell array having shortest route between each pair of DCs

% get the source Dc and destination DC
custdc = packarrevent(m,2);
suppdc = packarrevent(m,3);

if m == size(packarrevent,1)
    packarrevent = packarrevent(1:m-1,:);
elseif m == 1
    packarrevent = packarrevent(2:end,:);
else
    packarrevent = [packarrevent(1:m-1,:);packarrevent(m+1:end,:)];

% get the next package id
packid = packid + 1;

% get the time for next demand arrival
narrt = genrandnum(mrndn(custdc,suppdc),'exp');
nextarrivaltime = tnow+narrt;

% Schedule the arrival of next demand
ev = putnextevent(ev,[tnow+narrt packid 3]);

% put the information regarding the package scheduled to be generated in
% a separate matrix
node = length(routes{suppdc,custdc});
packarrevent = [packarrevent;packid custdc suppdc 0 node 0 0 0];
```

71
10. [pln.m] is a MATLAB function which provides information about PLN configuration.

```matlab
function [tp,TSI,W] = pln(pop,xy,IJD,TC,ts,p,lu,td,lf)
  %
  % [NT,AT,tp] = pln(pop,xy,IJD,TC,ts,p,lu,td)
  % pop - Population matrix(36X1) for each DC
  % xy - Location matrix(36X2) for each DC
  % IJD - Matrix containing the tail node, the head node and the distances between the
tail and the head nodes
  % TC - n-element Truck Capacity Vector
  % ts - Speed of the truck
  % p - m-element Proximity factor vector, contains different possible proximity
  % factors
  % lu - 1-element loading/unloading time vector, contains different possible loading
  % unloading time in minutes
  % td - Total demand (number of packages)
  % If - Load factor

  %Sort the matrices supplied
  TC = sort(TC);
  p = sort(p);
  lu = sort(lu);

  %Generate the demand at all DCs from the population around it.
  dem = (pop*100)/sum(pop);

  % Generate the demand from DC j at DC i
  w = dem*dem';
  dim = size(w,1);
  prox1{l1} = useProximity(p1,w,xy);
  tp = prox1{l1}*td;

  %Get IJT matrix for all loading and unloading time
  for j = 1:length(lu)
    IJT = IJD;
    IJT(:,3) = IJD(:,3)/ts;
    lut{j} = IJT;
  end

  %Get the demand matrix according to the proximity factor and the loading/unloading
time
  for l = 1:length(p)
    disp('Generating data...')
    for q = 1:length(lu)
      %Get all the routes
      A = full(list2adj(lut{q}));
      [D,P]=dijk(A);
      rte=pred2path(P);
    end
  end
```

72
45  %Get the demand for each arc in the network
46  w3 = zeros(dim,dim);
47  for i=1:dim
48     for j=1:dim
49         %Considers all the possible paths from P(pred2path)
50         b=rte{i,j};
51     end
52     if i == j
53         b = [b b];
54     end
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     prox{l,q} = w3;
61     end
62     end
63     W = w3;
64     m = 0;
65     for y = 1:length(p)
66         for z = 1:length(TC)
67             m = m + 1;
68             n = 0;
69             for s = 1:length(lu)
70                 n = n + 1;
71             end
72             %get the number of trucks
73             NT = genAvgTransTime(TC(z),prox{y,s},sum(sum(tp)),IJ,T,l);
74             end
75             end
76             %get the truck distribution for different DCs based on
77             %the demand
78             TSI = wselect(dem/100,1,ceil(sum(sum(NT))));
79             end
80             end
81             end
82             end
83             end
11. [simulation.m] is a MATLAB function which formats the input before starting the simulation.

```matlab
function [AT,WT] = simulation(nopack, trkcap, trksp, lu, pt, wup, runlen, s, IJD2, xy, pop, lf, pick, tp, TSI, W)
% [AT,WT] = simulation(nopack, trkcap, trksp, lu, pt, wup, runlen, s, IJD2, xy, pop, lf, pick, tp, TSI, W)
% AT = Average transportation time from simulation
% WT = Average waiting time from simulation
% nopack = total packages in the system per day
% trkcap = truck capacity
% trksp = truck speed
% lu = loadin/unloading time
% pf = proximity factor
% wup = simulation warm-up period
% runlen = simulation run length
% s = seed in simulation
% IJD2 = IJD matrix for simulation
% xy = XY coordinate of each DC
% pop = population around each DC
% lf = load factor
% pick = way a queue of packages is picked
% tp = total package deand between each pair of DCs
% TSI = vector having DC id assigned to each truck
% W = link flow

TK.trkspeed = trksp; % Truck speed
TK.mxtrkcap = trkcap; % Truck max. capacity
TK.loadfact = lf; % Load factor
TK.tsi = TSI; % Truck distribution at different DCs

PK.total = nopack; % Total package in the system per day
PK.tp = tp; % Total packages between different DCs
PK.w = W; % Packages for each route

DC.XY = xy;
DC.Pop = pop;
DC.lut = lu; % Loading/unloading time at DC
DC.prxf = pf; % Proximity factor
DC.ppt = pt; % Process time for demand

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

[AT,WT] = startsimulation(DC, TK, PK, SIM, IJD2, pick);
```
12. [start.m] is a MATLAB function where the input is provided for the simulation.
1   function start
2
3   %load data related to PLN having 36 DCs
4   load plnx36;
5   %truck capacity
6   trkcap = [20];
7   % number of packages per day in the system
8   nopack = [5000];
9   %truck speed
10  trkspeed = [60];
11  %loading/unloading time
12  lut = [0];
13  %proximity factor
14  pf = [0];
15  %processing time in DC
16  pt = 0;
17  %simulation warm-up period
18  warmpperiod = 0;
19  %simulation run length
20  runlength = 1000;
21  %load factor
22  If = [0.8];
23  %1 for maximum value or number queue, 0 for weighted selection, 2 for the scheduled version
24  qpick = [1];
25
26  for qp = 1:length(qpick)
27     for p = 1:length(seed)
28        for i = 1:length(nopack)
29           for j = 1:length(trkcap)
30              for k = 1:length(trkspeed)
31                 for m = 1:length(lut)
32                    for n = 1:length(pf)
33                       for q = 1:length(If)
34                          %For PLN
35                          [tp, TSI, W] = pln(DC.Pop, DC.XY, JD2, trkcap(j), trkspeed(k), pf(n), lut(m), nopack(i), If(q));
36                          %For simulation
37                          [ATT, WT] = simulation(nopack(i), trkcap(j), trkspeed(k), lut(m), pf(n), pt, warmpperiod, runlength, seed(p), JD2, DC.XY, DC.Pop, If(q), qpick(qp), tp, TSI, W);
38                   end
39               end
40             end
41           end
42         end
43     end
44   end
45 end

75
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          %initialize global variables
18          [ts,lut,ppt,out,td,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK);
19          
20          %Initialize random numbers
21          [mrndn] = initializerandomnumbers(tp);
22          
23          %Initialize other objects in simulation
24          [ev,trak,pdc,packarrevent,allpackids,packid,T,routes] = initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,pick,W);
25          
26          %For statistics collection and plotting graphs
27          si = size(tp);
28          intArrMean = zeros(si);
29          intArrVar = zeros(si);
30          intArrNum = zeros(si);
31          intArrOld = zeros(si);
32          loadMean = zeros(si);
33          loadVar = zeros(si);
34          loadNum = zeros(si);
35          waitMean = zeros(si);
36          waitVar = zeros(si);
37          waitNum = zeros(si);
38          
39          out = 0;
40          count = 0;
41          totpack = 0;
42          totval = 0;
43          totval2 = 0;
44          
45          avgtrkld = [0 0];
46          avgtt = [0 0];
47          

avgdcl = [0 0];
avgwt = [0 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 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);
    end
    %Get the DC id at which truck arrives
    if get(trk(tr),'cdn') == 0
        %for trucks coming from other DC
        currdcid = get(trk(tr),'ndcn');
        trk(tr) = set(trk(tr),currdcid,'cdn');
        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),'cdn');
    end
end

%get the packages
package = get(trk(tr),'data');

%get the number of packages in the truck
numpack = size(package,1);

if numpack > 0  %if number of packages on the truck is more than zero, unload them
    trk(tr) = set(trk(tr),[],'data');</span>
  [trk,pdc,rd,td,avgwt] = unloadpackages(tr,currdcid,trk,pdc,rd,td,tnow,avgwt,package,routes);
end

%set the truck in the queue at a DC
pdc(currdcid) = set(pdc(currdcid),tr,'truckid');</span>

%get the next DC for which there is a queue with maximum value
maxnumpack = max(get(pdc(currdcid),'packsuppbay'));

%if there are any packages lying in the DC
if maxnumpack ~< 0
    [trk,pdc,ev,W1,ve,waitMean,waitVar,waitNum] = loadpackages(currdcid,ev,trk,pdc,tnow,lut,pick,ve,routes,W,waitMean,waitVar,waitNum);
end

elseif r(3) == 0  %truck leaving a DC

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

%set the final property of the truck before its departure and
%put the next event of the arrival of the truck, at the next DC, in the event calendar
cdc = get(trk(tr),'cdcn');
ndc = get(trk(tr),'ndcn');

transtime = T(cdc,ndc);

if tnow > 4
    %For Mean and variance of inter-arrival of trucks
    oldMean = intArrMean(cdc,ndc);
    intArr = tnow-intArrOld(cdc,ndc);
    intArrMean(cdc,ndc) = (intArrMean(cdc,ndc)*intArrNum(cdc,ndc) + intArr)/(intArrNum(cdc,ndc)+1);
    if intArrNum(cdc,ndc) >= 2
        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);
    elseif intArrNum(cdc,ndc) == 1
        intArrVar(cdc,ndc) = oldVar + (oldMean-intArr)^2/2 + 1/12;
intArrVarTemp = intArr;
intArrVar(cdc, ndc) = power(power(intArrMean(cdc, ndc) - intArrVarTemp, 2) + power(intArrMean(cdc, ndc) - intArrVar(cdc, ndc), 2), 0.5);
else if intArrNum(cdc, ndc) == 0
    intArrVar(cdc, ndc) = intArr;
end
intArrNum(cdc, ndc) = intArrNum(cdc, ndc) + 1;
% For mean and variance of truck load
oldMean = loadMean(cdc, ndc);
load = get(trk(tr), 'currnumpack');
loadMean(cdc, ndc) = (loadMean(cdc, ndc)*loadNum(cdc, ndc) + load)/(loadNum(cdc, ndc)+1);
if loadNum(cdc, ndc) >= 2
    loadVar(cdc, ndc) = power((power(loadVar(cdc, ndc), 2) + loadNum(cdc, ndc) - 1), 0.5);
else if loadNum(cdc, ndc) == 1
    loadVar(cdc, ndc) = load;
end
loadNum(cdc, ndc) = loadNum(cdc, ndc) + 1;
end
intArrOld(cdc, ndc) = tnow;
trk(tr) = set(trk(tr), 0, 'cdn');
ev = putnextevent(ev, [tnow + transtime, tr, 1]);
% Processed package arriving
else if r(3) == 3
    ev = putnextevent(ev, [tnow, r(2), 4]);
% Package demand arriving
else if r(3) == 4
    get the package id
    pid = r(2);
    allpack = packarrevent(:, 1);
    allpack(allpack == pid) = 0;
    m = find(allpack);
    % Get the source DC and destination DC
    currdc = packarrevent(m, 3);
    destdc = packarrevent(m, 2);
    if currdc == destdc
        rd = rd + 1;   % To satisfy local demand
else
  packarrevent(m,4) = 1;
  packarrevent(m,6) = r(1);
  packarrevent(m,7) = 0;
  packarrevent(m,8) = r(1);

  % check if trucks are available at the DC
  numoftrucksids = get(pdc(currdc),'truckid');

  ndc = routes{currdc,destdc}(2);
  if pid == 4000
    fprintf('coming out.
');
    routes {currdc,destdc}
  end
  % if no trucks are available at the DC
  if size(numoftrucksids,2) == 0
  end

  % put the package in a queue in the DC
  pdc(currdc) = set(pdc(currdc),packarrevent(m,:),'data');
  pdc(currdc) = set(pdc(currdc),get(pdc(currdc),'currnumpack')+1,'currnumpack');
  dcs = get(pdc(currdc),'connecteddcs');
  dcdst = dcs;
  dcdst(dcdst ~= ndc) = 0;
  j = find(dcdst);

  pdc(currdc) = set(pdc(currdc),j,j,1,'numpacksuppbay');
  pdc(currdc) = set(pdc(currdc),j,j,1,'packsuppbay');

  % if truck is available
  else
    % put the package in the truck and put the event of the departure of the truck in the event calendar.
    truc = numoftrucksids(1);
    trk(truc) = set(trk(truc),packarrevent(m,:),'data');
    trk(truc) = set(trk(truc),1,'currnumpack');
    trk(truc) = set(trk(truc),ndc,'nden');
    pdc(currdc)= removetruck(pdc(currdc),truc);
    ev = putnextevent(ev,[tnow,truc,0]);
  end
  end

  % Schedule the arrival of the next package for the same DC combination
  [packarrevent,packid,ev] = nextarrival(packarrevent,packid,mrdn,ev,IJD2,tnow,m,routes);

  elseif r(3) == 5  % draw the graph
    avgload = 0;
    avgdload = 0;
    avgwttime = 0;
for j = 1:length(trk)
    avgload = avgload + get(trk(j), 'currnumpack');
end

for km = 1:length(pdc)
    avgdcload = avgdcload + get(pdc(km), 'currnumpack');
end

tnow = r(1)
avgwtime = avgwt/rd
ATT = td/rd

if r(3) == -1  % end of simulation
    break;
end

elseif r(3) == -2  % Warm up period ends
    td = 0;
    rd = 0;
    avgwt = 0;
    end

if out == 1
    break;
    end

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

    function [trk,pdc,rd,td,avgwt] = unloadpackages(tr,currcid,trk,pdc,rd,td,tnw,avgwt,pkg,routes)
    
    %
    %[trk,pdc,rd,td,avgwt] = unloadpackages(tr,currcid,trk,pdc,rd,td,tnw,avgwt,pkg,routes)
    %trk = array of truck objects
    %pdc = array of plndc objects
    %rd = number of packages reached their destinations
    %td = current average transportation time
    %avgwt = current average waiting time
    %tr = truck id
    %currcid = current DC id
    %pkg = array containing information about the packages being unloaded
    %routes = array cell array having shortest distance between each pair of DCs

    %increase the number of intermediate DC by 1
    pkg(:,4) = pkg(:,4) + 1;

    %get the packages whose final destination is not this DC
    ptktemp = pkg(:,[4 5]);
    ptktemp(ptktemp(:,1) == ptktemp(:,2)) = 0;
    mnr = find(ptktemp(:,1));
    mnr pkg = pkg(mnr,:);

    %get the packages whose final destination is this DC
    ptktemp = pkg(:,[4 5]);
    ptktemp(ptktemp(:,1) == ptktemp(:,2)) = 0;
    mr = find(ptktemp(:,1));
    mr pkg = pkg(mr,:);

    if size(mr,1) > 0 %package reached the destination
    %increase the total number of packages reached the destination by 1
    rd = rd + size(mr pkg,1);
    %set the new total transportation time
    td = td + size(mr pkg,1)*tnw - sum(mr pkg(:,6));
    %set the average waiting time
    avgwt = avgwt + sum(mr pkg(:,7));
    end

    %If this is the intermediate DC
    if size(mnr,1) > 0
    mnr pkg(:,8) = tnow;
    %set the packages in the DC
    pdc(currcid) = set(pdc(currcid),mnr pkg,'data');
    %set the variables for DC
    pdc(currcid) =
    set(pdc(currcid),get(pdc(currcid),'currnumpack')+size(mr,1),'currnumpack');
    end

72
dcs = get(pdc(currdcid),'connecteddc');
%increase the number of packages in the respective bays where packages are stored to be
%transported to their next DCs
for i = 1:size(mnr,1)
  ndc = routes{mnrpkg(i,3),mnrpkg(i,2)}(mnrpkg(i,4)+1);
dest = dcs;
dest(dest ~= ndc) = 0;
j = find(dest);
pdc(currdcid) = set(pdc(currdcid),1,j,1,'numpacksuppbay');
pdc(currdcid) = set(pdc(currdcid),1,j,1,'packsuppbay');
end
%set the number of packages in the truck to 0
trk(tr) = set(trk(tr),0,'currnumpack');
14. [useProximity.m] is a MATLAB function which helps in demand generation based on proximity factor.

```matlab
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.

```matlab
function [furdc,tr,out,nstepn] = bidding(currdcid,routes,T,tnow,varargin)

% [furdc,tr,out,nstepn] = bidding(currdcid,routes,T,tnow,varargin)
%furdc = vector having dc ids where load formation changes because of reneging
% tr = truck id
%out = cell array having Information about the packages that are in loads
%nstepn = containing dc ids where load formation changes because of reneging by a truck
%currdcid = current dc id
%routes = cell array having shortest route for each pair of DCs
%T = Travel time between each directly connected DCs
%now = current simulation time
%varargin = parameter list
%varargin{1} = iteration number
%varargin{2} = not used currently

global trkinf;
global pdc;
global trk;

out = {};
outr = {};
outt = {}; 
selected = 0;
furdc = [];
tr = [];
nstepn = {};
e = 0;

%Get the current DC
cd = pdc(currdcid);

%get the trucks in the increasing order of distance from the DC
iter = varargin{1};

if iter == 1
    ns = varargin{2};
end

if iter == 1
    trkinfotemp = trkinf(:,2);
    trkinfotemp(trkinfotemp == currdcid) = 0;
    temptime = trkinf(:,1);
    temptime(temptime < tnow) = tnow;
```

85
temptime = temptime + dijk(T, trkinfo(:,2), currdcid);

temp = [temptime trkinfotemp trkinfo(:,6)];

[t,idx] = sortrows(temp);

for i = 1:length(pdc)
    pdc(i) = set(pdc(i), 'info');
end

% For without bidding
trkinfotemp = trkinfo(:,2);
trkinfotemp(trkinfotemp == currdcid) = 0;
temptime = trkinfo(:,1);

if temptime < tnow = tnow;

tramptime = temptime + dijk(T, trkinfo(:,2), currdcid);

temp = [temptime trkinfotemp trkinfo(:,6)];

% For without protocol
trktimetemp = temp(:,1);
trktimetemp(trktimetemp - tnow > 0.00001) = 0;

trktimetempin = find(trktimetemp);

temp = temp(trktimetempin,:);

trkdctemp = temp(:,2);
trkdctemp(trkdctemp == 0) = -1;

trkdctempin = find(trkdctemp);

temp = temp(trkdctempin,:);

% End of addition
[t,idx] = sortrows(temp);

% For without protocol

currdcid

trkinfo

idx = t(:,4);

idx

% End of addition

elseif iter <= 1

idx = varargin{2};

conndc = varargin{3};

end

for k = 1:length(idx)

out = {};

nstepn = {};

if iter == 1
% get the load information from the current DC
[outr, don, nstpt, e] = biddingmult(currdcid, routes, T, tnow, idx(k), ns);

if size(outr, 1) == 0
    break;
end
pkgsm = outr{1,4} ;
if size(pkgsm, 1) == 0
    continue;
end
if don == 1
    break;
elseif don == 2
    continue;
end
% Get load information from intermediate DCs along the intended path to the current DC
outr = biddingsingle(currdcid, routes, T, tnow, iter, idx(k), conndc);
end
bidtruck = idx(k);
dcont = get(trk(idx(k)), 'dcont' );
currdcid;
end
r = routes{trkinfo(idx(k), 2), currdcid};
if r(end-1) == r(end)
    if size(outr, 1) > 0
        if outr{1,1} ~= 0
            [furdc, tr, outr] = bidding(r(end-1), routes, T, tnow, iter+1, idx(k), currdcid);
        end
    end
    end
    end
    out = [outr; outr; outr];
if iter == 1
    if size(out, 1) > 0
        % If there is a load, see if it results in reneging. If yes, change the configuration of
        % the affected trucks and DCs
        [furdc, tr, done, nstpt, ind, cvimp] = changeConfig(out, tnow, T);
        if e == 1
            if 0 - cvimp > 0.00001
                break;
            end
        end
        if done == 1
            if ind ~= 0
                if size(nstpt, 1) > 0
                    nstpt{ind, 4} = [nstpt{ind,4}; nstpt{1,4} ];
                end
            end
            else
                nstpt = [nstpt; nstpt];
            end
        end
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
1 function [out,don,nstep,t,e] = biddingmult(currucid,routes,T,tnow,tr,ns)
2 %
3 %[out,don,nstep,t,e] = biddingmult(currucid,routes,T,tnow,tr,ns)
4 %out = cell array having information about the packages in loads at current DC
5 %don = flag to check if there are packages in the DC
6 %nstepn = cell array having information about the packages in the load which helps in
7 %further iterations to form next load
8 %e = not used currently
9 %currucid = current DC id
10 %routes = cell array having shortest route between each pair of DCs
11 %T = Matrix having journey time between two directly connected DCs
12 %tnow = current simulation time
13 %tr = truck id
14 %ns = not used currently
15 global trkinfo;
16 global pdc;
17 global trk;
18
19 nstep = {};  
20 e = 0;
21
22
23 %Get the current DC and the information of packages (at the DC and incoming)
24 cd = pdc(currucid);
25 dcinfo = get(cd,'info');
26 dcinfo2 = size(dcinfo,2);
27 dcinfo2 = cell(1,dcinfo2);
28 out = {}; 
29 don = 0;
30 con = 0;
31 firsttime = [];
32 y = 0;
33
34 %If there are no packages at a DC
35 if isequal(dcinfo,dcinfo2) == 0
36
37
38 n = tr;
39 idx = tr;
40
41
42 ndcids = get(cd,'connecteddc');
43 packsnew = [];
44 packsold = [];
45 dcidnew = 0;
46 dcidold = 0;
47 trkid = 0;
bidtemp = []; 
nopp = 0; 
totbidfinal = 0; 
selected = 1; 
m = 0; 

%Get the times at which different trucks are reaching at the current DC 
times = get(cd,'incomingtrucks'); 
if size(times,1) == 0 
times = tnow; 
else 
times = [tnow;times(:,2)]; 
end 

%If a truck is already at the DC, make its time of arrival as the current simulation time 
times((tnow-times) > 0.00001) = tnow; 
times = unique(times); 

ndci = 1:length(ndcids); 

for k = 1:length(t) 
trkcap = get(trk(idx(k)),'maxcapacity'); 
indexndc = 0; 
fl = 0; 
timepack = 0; 
dcconts = get(trk(idx(k)),'dcont'); 
[dccont,mem] = ismember(dcconts,currcid); 
dccontin = find(dccont); 

if length(dccontin) ~= 0 
y = dccontin; 
end 

%time for truck to reach the current DC 
timetake = dijk(T,trkinfo(idx(k),2),currcid); 
trktime = trkinfo(idx(k),1); 
trktime(tnow-trktime > 0.00001) = tnow; 
tottimetake = trktime+timetake; 

if y == 0 
con = 1; 
else 
con = 2; 

elseif 
%If truck has reservation at the current DC, time at which truck would pick the load 
if y < length(dccont) 
firsttime = tottimetake;
100  elseif y == length(dccont)
101     firsttime = trkinfo(tr,8);
102   end
103   end
104
105  %If truck has reservation at the current DC, put those packages for load formation
106  if length(dccont) > 0
107     indexdcontc = get(trk(idx{k}),'indexdcontd');
108     indexdnc = indexdcontc(dccont);
109  end
110
111  %form cell arrays and matrix to store information about load formation
112  numofindex = zeros(length(times),length(ndci));
113  index = cell(length(times),length(ndci));
114  bids = cell(length(times),length(ndci));
115  totbid = zeros(length(times),length(ndci));
116  indexfinal = cell(length(times),length(ndci));
117  bidsfinal = cell(length(times),length(ndci));
118  pkgtime = cell(length(times),length(ndci));
119
120  %Form various loads
121  for i = 1:length(times)
122     for jk = 1:length(ndci)
123        j = ndci(jk);
124        if size(dcinfo{j},1) > 0
125           dcinfof = dcinfo{j};
126           time = dcinfof(:,3);
127           if abs(times(i) - tnow) < 0.00001
128              time(time - tnow > 0.00001) = 0;
129           elseif tnow - times(i) > 0.00001
130              time(time - tnow > 0.00001) = 0;
131           else
132              time(time - times(i) > 0.00001) = 0;
133           end
134           in = find(time);
135           bids{i,jk} = dcinfof(in,2);
136           [bids{i,jk},ind] = sort(bids{i,jk});
137           index{i,jk} = in(ind);
138           pkgtime{i,jk} = dcinfof(index{i,jk},3);
139        end
140    end
141
142    end
\% fb = bids\{i,j-1\};
\% ft = pkgtime\{i,j-1\};
\% fi = index\{i,j-1\};
\% fb = bids\{i,jk\};
\% ft = pkgtime\{i,jk\};
\% fi = index\{i,jk\};
\% if length(fb) > 0
\% [ub,ui] = unique(fb);
\% if length(ub) == length(fb)
\% iin = 1;
\% fin = 1;
\% for u = 1:length(ui)
\% fin = ui(u);
\% [ftt,fit] = sort(ft([iin:fin]));
\% ft(end-1:1) = fit(1:end);
\% ft(end-1:1) = fit(1:end);
\% ftt = fit([iin:fin]);
\% ftt = ftt(fit);
\% ft(iin:fin) = fit;
\% fi(iin:fin) = ftt;
\% iin = ui(u)+1;
\% end
\% end
\% index\{i,jk\} = fi;
\% pkgtime\{i,jk\} = ft;
\% end
\% for i = 1:size(bids,1)
\% for j = 1:size(bids,2)
\% b = bids\{i,j\};
\% if length(b) > 0
\% if length(b) > trkcap
\% bidsfinal\{i,j\} = [sum(b(end-trkcap+1:end))];
\% indexfinal\{i,j\} = index\{i,j\}(end-trkcap+1:end);
\% else
\% bidsfinal\{i,j\} = [sum(b(1:end))];
\% indexfinal\{i,j\} = index\{i,j\}(1:end);
\% end
\% numofindex(i,j) = length(indexfinal\{i,j\});
\% if length(bidsfinal\{i,j\}) <= 0
\% totbid\{i,j\} = bidsfinal\{i,j\};
else
totbid(i,j) = 0;
end
end

[bidmax,bidind] = max(totbid,[],2);
end
ttr = tottimetake;
if ttr - times(end) > 0.00001
 e = 1;
else
e= 0;
end
timet = times;
if fl == 1
timet((timet - ttr) > 0.00001) = 0;
ind = find(timet);
if length(ind) == 0
 ind = 1;
else
 ind = ind(end);
end
e
end

out = {}
out = cell(1,10);
m = m + 1;
out{m,1} = idx(k);
out{m,3} = currdcid;
out{m,5} = [];
out{m,6} = [];
out{m,7} = [];
out{m,10} = [];
out{m,11} = firsttime;

mn = 0;
for k = ind:length(times)
out2t = [];
out8t = [];
out9t = [];
packarray = {}
for index = 1:size(totbid,2)
out2t = [out2t ndci(index)];
finalindex = indexfinal{k,index};
if size(dcinfo{1,ndci(index)},1) > 0
dcinfo1 = dcinfo{1,ndci(index)};
dcinfoi2 = dcinfoi(finalindex,:);
packarray{1,index} = dcinfoi2;
else
packarray{1,index} = [];
end
out8t = [out8t totbid(k,index)];
end
out{m,2} = [out{m,2};out2t];
out{m,8} = [out{m,8};out8t];
out{m,4} {k-mn-ind+1,1} = packarray;
if fl ~= 1
if ttr - times(k) > 0.00001
out{m,10} = [out{m,10} ttr];
else
out{m,10} = [out{m,10} times(k)];
end
else
out{m,10} = [out{m,10} times(k)];
end
end
end
end

%If truck has reservation with the current DC, put a flag in the output
%of this function
if currdcid == dcconts(dccontin)
out{m,9} = [dccontin indextonde(dccontin)];
end
selected = 1;
end
pdc(currdcid) = cd;
else
don = 1;
end
3. [biddingsingle.m] is a MATLAB function which simulates bidding in the intermediate DCs.

```matlab
function out = biddingsingle(currdcid, routes, T, tnow, iter, tr, conndc)
% out = biddingsingle(currdcid, routes, T, tnow, iter, tr, conndc)
% Informaiton about the load
% currdcid = current DC id
% routes = cell array having shortest route for each pair of DCs
% T = Travel time between each directly connected DCs
% tnow = current simulation time
% iter = iteration number
% tr = truck id
% conndc = list of directly connected DCs

global pdc;
global trk;
global trkinfo;

index = [];
bids = [];
bidsfinal = 0;
indexfinal = [];
out = {};
indexdc = 0;
f1 = 0;

% get object of current DC
cd = pdc(currdcid);

% get the index to next dc id
cdc = get(cd,'connecteddc');
nextid = cdc;
next = cdc;
nextd = next;
nexti = nextd;

% get the time taken by truck to reach the DC
timetake = dijk(T, trkinfo(tr, 2), currdcid);
ttime = trkinfo(tr, 1);
if tnow - curtime > 0.00001
  curtime = tnow;
end
tottimetake = curtime + timetake;

% check if truck already has reservation with the DC
dcct = get(trk(tr), 'decont');
dccont = ismember(dcct, currdcid);
dccont = find(dccont);

%If truck has reservation with the DC, put the package it has reserved in load formation
if length(dccont) > 0
```

95
indextondc = get(trk(tr), 'indextonextdc');
indexdc = indextondc(dcontin);
if indexdc == ndin
  if dcontin(1) == length(dconts)
    if trkinfo(tr,8)-tottimetake > 0.0001
      fl = 1;
    else
      fl = 1;
    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(dcontin) > 0
  if dcontin(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}(:,2);
    [bids,i] = sort(bids);
    index = in(i);
    trkcap = get(trk(tr), 'maxcapacity');
  end
end

96
b = bids;
if length(b) > 0
if length(b) > trkcap
bidsfinal = [sum(b(end-trkcap+1:end))];
indexfinal = index(end-trkcap+1:end);
else
bidsfinal = [sum(b(1:end))];
indexfinal = index(1:end);
end
totbid = bidsfinal;
out{1} = tr;
out{2} = ndin;
out{3} = currdcid;
out{4} = dcinfo{ndin}(indexfinal,:);
out{5} = [];
out{6} = [];
out{7} = [];
out{8} = totbid;
out{9} = [];

% if there are no packages at the DC
else
out{1} = tr;
out{2} = ndin;
out{3} = currdcid;
out{4} = [];
out{5} = [];
out{6} = [];
out{7} = [];
out{8} = 0;
out{9} = [];
end
end
out{10} = [];
out{11} = [];
4. [createdc.m] is a MATLAB function which creates PLN DC objects.

```matlab
function c = createdc(n, IJD2, trk, W, pick)
%
% c = createdc(n, IJD2, trk, W, pick)
% c = array of plndc object
% n = number of DCs in a PLN
% IJD2 = IJD matrix for a PLN
% trk = a vector of truck objects
% W = line flow matrix
% pick = flag to pick a queue of packages
seq = [];
A = full(list2adj(IJD2));
disp('Generating DC...');
for i = 1:n
%
% get the list of connected DCs
conn = [i];
for j = 1:size(A,2)
if A(i, j) ~= 0
conn = [conn j];
end
end
if pick == 2
W1 = W(i, conn(2:end));
seq = fixseq(W1);
end
%
% set the number of packages in each bay as 0
suppbay = zeros(size(conn, 1), size(conn, 2));
%
% set the package ids as null
pid = [];
%
% set the truck ids as null
tid = [];
%
% for the DC object
(i) = plndc(i, inf, 0, conn, suppbay, suppbay, pid, tid, [], seq, {}, {}, [], 0);
end
```
5. [creatrucks.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) =
10           truck(i, TSI(i), TSI(i), {}, 0, TK.mxtrkcap, 0, TSI(i), TSI(i), TSI(i), 0, 0, 0, 0, TK.trkamt);
11    end
6. [genAvgTransTime.m] is a MATLAB function which gives association of trucks with DCs in the beginning of the simulation.

```matlab
1  %NT = genAvgTransTime(mct,pcd,IJT,td,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.

```matlab
function [ran] = genrandnum(varargin)
% [ran] = genrandnum(varargin)
% ran = random number
% varargin = list of parameters
% varargin{1} = mean of a distribution
% varargin{2} = name of the distribution
% varargin{3} = a flag to generate demand based on time
r = [];
if length(varargin) == 2
switch varargin{2}
% For exponential distribution
case 'exp'
    r = exprnd(varargin{1},1,1);
% For uniform distribution
case 'unif'
    r = unifrnd(0,50,1,1)*20;
end
elseif length(varargin) == 3
fl = varargin{3};
switch varargin{2}
% For time based uniform distribution
    case 'unif'
        if fl == 1
            r = unifrnd(0,50,1,1)*20;
        elseif fl == 2
            r = unifrnd(0,50,1,1)*20*5;
        elseif fl == 3
            r = unifrnd(0,50,1,1)*20/5;
        end
end
ran = r(1,end);
```
8. [initializesimulation.m] is a MATLAB function which initializes the simulation.

```matlab
1 function [ev,trk,pdc,packarrevent,allpackids,packid,T,routes,trkinfo] = 
2 initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,packid,W,PK,B)
3 %
4 %[ev,trk,pdc,packarrevent,allpackids,packid,T,routes] = 
5 initializesimulation(tp,IJD2,ts,packid,mrndn,TSI,SIM,TK,packid,W)
6 %ev = object for event calendar
7 %trk = array for truck object
8 %pdc = array for plndc objects
9 %packarrevent = array containing package information
10 %T = matrix having time between each connecting DCs
11 %routes = cell array for the shortest route between two DCs
12 %tp = demand for package for each pari of DCs
13 %IJD2 = IJD matrix for a PLN
14 %mrndn = matrix having means to generate demand for each pari of DCs
15 %TSI = vector having DC ids for different trucks at the start of the
16 %simulation
17 %SIM = values for the simulation
18 %TK = information to create truck objects
19 %pick = way a queue can be picked at a DC
20 %W = link flow
21 %PK = Information about packages
22 %B = vector having package values
23 global pdc;
24 A = full(list2adj(IJD2));
25 T = A/ts;
26 dim = size(A,1);
27 packid = 0;
28 %Matrix having number of packages for each pair of DCs
29 wa = PK.ar/10000;
30 wa = ceil(wa*PK.total);
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 %create event calendar
40 ev = eventcal;
41 %create DCs
42 pdc = createdc(dim,IJD2,trk,W,pick);
43 %For package traking in event calendar
```
packarrevent = [];
nt = length(trk);
tnow = 0;
disp('putting trucks arrival in the event calendar....');
for i = 1:nt
  %1 for arrival of trucks at a DC
ev = putnextevent(ev,[0.1 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;
  %nrarr = genrandnum(1/mrndn{i,j}(1),'exp');
  %ev = putnextevent(ev,[tnow+nrarr,packid,3]);
  %nodec = length(routes{j,i});
  %packarrevent = [packarrevent,packid i j 0 node 0 0 0 0 0 0];
end

BV = B;
for i = 1:size(wa,1)
  for j = 1:size(wa,2)
    if i ~= j
      for k = 1:wa(i,j)
        packarr = [];
        packid = packid + 1;
      end
      %get the value of package
      bidamt = BV(packid);
    end
    %Put the packages under 'info' for DC
dis = dijk(A,i,j);
    bidamt = bidamt*dis;
    ndc = routes{j,i}(2);
    condc = get(pdc(j),'connecteddc');
    [x,y] = ismember(ndc,condc);
    %make packarrevent for the packages
    nodec = length(routes{j,i});
    packarr = [packid i j 1 nodec 0 0 0 0 bidamt 0 0];
    packarr(9) = packarr(10)*A(j,ndc)/dijk(A,packarr(3),packarr(2));
    pdc(j) = set(pdc(j),[packid packarr(9) 0.05],y,'infot');
    pdc(j) = set(pdc(j),packarr,'data');
    pdc(j) = set(pdc(j),[packid 0],'packtrk');
```plaintext
97 end
98 end
99 end
100 end
101 ev = putnextevent(ev,[SIM.wp -2 -2]);
102 ev = putnextevent(ev,[SIM.rl -1 -1]);
103 [ev,r] = getnextevent(ev,0);
104 % Add for truck location information
105 trkinfo = zeros(length(trk),11);
106 trkinfo(:,4) = TK.trkamt;
107 for i = 1:length(trk)
108 trkinfo(i,2) = get(trk(i),'cdcn');
109 pdc(trkinfo(i,2)) = set(pdc(trkinfo(i,2)),[i 0],'incomingtrucks');
110 end
```
9. [initializevariables.m] is a MATLAB function which initializes the variables in the simulation.

```matlab
function [ts,lut,ppt,out,rd,packid,tp,TSI,W] = initializevariables(TK,DC,PK)
%TK = information to create truck objects
%DC = information to create dc objects
%PK = information to create package objects
%Truck Speed
%Loading/Unloading time
lut = DC.lut;

%Processing time for packages
pmt = DC.pmt;

%break variable
out = 0;

%Number of observations
rd = 0;

td = 0;

%package id
packid = 0;

%total package demand between different DCs
tp = PK.tp;

%Truck distribution at different DCs
TSI = TK.tsi;

%percentage package distribution for each route
W = PK.w;
```
10. [initializerandomnumbers.m] is a MATLAB function which initializes the random numbers in the simulation.

```matlab
1  %This function initialize the variables used to generate random numbers.
2  function [mrndn] = initializerandomnumbers(tp)
3    %
4    %[mrndn] = initializerandomnumbers(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.

```matlab
function [ev] = loadpackages(currccd, ev, tnow, lut, ve, routes, W, waitMean, waitVar, waitNum, A, varargin)
    %
    %[trk, pdc, ev, W1, ve, waitMean, waitVar, waitNum] = loadpackages(currccd, ev, trk, pdc, tnow, lut, pick, ve, routes, W, waitMean, waitVar, waitNum)
    % ev = object for event calander
    % currccd = current DC id
    % tnow = current simulation time
    % lut = loading/unloading time
    % ve = not is use
    % routes = cell array habing shortest distance between each pair of DCs
    % W = link flow
    % waitMean = runnign average of the waiting time
    % waitVar = running variance of waiting time
    % waitNum = number of observations to calculate current mean waiting time
    % A = matrix having distance between two directly conected DCs
    % varargin = list of parameters
    % varargin{1} = truck id
    % varargin{2} = list of pack id to be loaded on the truck
    global trkinfo;
    global trk;
    global pdc;
    global allcosts;
    oldbidamt = 0;
    T = A/60;
    pkgleave = [];
    pval = 0;
    %Get the current DC
    cd = pdc(currccd);
    pksid = get(cd,’data’);
    leave = 1;
    trkid = varargin{1};
    ndcid = 0;
    if length(varargin) == 2
        pks = [];
        %List of packages to be loaded on the truck
        packset = varargin{2};
        if size(packset,1) > 0
            %Check if all the packages in the list are at the DC
            packs = packset(:,1);
            if size(pksid,1) > 0
```

107
pks = pkSID(:,1);
end
c = ismember(pks,packs);
pksIndAv = find(c);

% If all the packages in the list are not at the DC, do not load the
% packages
if length(pksIndAv) == length(packs)
    leave = 0;
pkgleave = [];
pksidtemp = pkSID;
else
    pkgleave = pksid(pksIndAv,:);
    ndcid = routes{pkgleave(1,3),pkgleave(1,2),(pkgleave(1,4)+1);
    pkgIndNotLeave = setdiff([1:length(pks)],pksIndAv);
pksidtemp = pkSID(pkgIndNotLeave,:);
[cd,oldbidamt] = updateTruck(ntrkid,cd);
end
% If list is empty, change the configuration of truck and DC directly
else
    trknxtDcs = get(trk(ntrkid),'dcont');
    ndcid = trknxtDcs(2);
pksidtemp = pksid;
    leave = 1;
end

% change the configuration of truck
[cd,oldbidamt] = updateTruck(ntrkid,cd);
end
cdcs = get(cd,'connectedDcs');
nd = cdcs;
ndc = ndcid;
nd(nd ~= ndc) = 0;
ndin = find(nd);
am = ndin;

if size(pkgleave,1) > 0
% If number of packages leaving a DC is greater than zero
% set the statistics for the package to be shipped
pkgleave(:,7) = pkgleave(:,7) + (tnow - pkgleave(:,8));
totbidamtall = sum(pkgleave(:,9));
trkinfo(ntrkid,3) = trkinfo(ntrkid,3) - oldbidamt;
trkinfo(ntrkid,9) = trkinfo(ntrkid,9) - (totbidamtall - pval);

% set the variables for truck
trk(ntrkid) = set(trk(ntrkid),size(pkgleave,1),'currnumpack');
%set the variables for DC
cd = setcd(size(pkstemp,1),’currnumpack’);

%set the variables for DC
if size(pkstemp,1) > 0
pkgtrk = getcd(),packtrak’;
[x,y] = setdiff(pkgtrk,1),pkstemp,1));
if length(find(y)) > 0
    cd = setcd(pkgtrk(y,:),’setpacktrak’);
else
    cd = setcd,[],’setpacktrak’);
end
e else
end

%remove the truck from the queue
trkinfo(ntrk,10) = trkinfo(ntrk,10) - tcs;
trk(ntrk) = settrk(ntrk),ndc,‘ndc’);

%remove the truck from the queue
ind = setdiff([1:length(incoming)],in);
cd = set(cd, incomingt(ind,:), 'setincomingtrucks');
%schedule the departure of the truck
ev = cancelEvent(ev, [ntrkid 0]);
ev = putnextevent(ev, [tnow + lut / 60, ntrkid, 0]);
transtime = T(currclid, nclid);
trkinfо(ntrkid, 1) = tnow + transtime + 2 * lut;
trkinfо(ntrkid, 2) = nclid;
% Add to new dc
trkdc = trkinfо(., 2);
trkdc(trkdc == currclid) = 0;
trkdcin = find(trkdc);
trktum = trkinfо(trkdcin, 1);
[trktum, indt] = sort(trktum);
trkdcinfі = trkdcin(indt);
trkinfо(trkdcinfі, 6) = [1:length(trkdcinfі)]';
% Remove from old dc
trkdc = trkinfо(., 2);
trkdc(trkdc == currclid) = 0;
trkdcin = find(trkdc);
trktum = trkinfо(trkdcin, 1);
[trktum, indt] = sort(trktum);
trkdcinfі = trkdcin(indt);
trkinfо(trkdcinfі, 6) = [1:length(trkdcinfі)]';
trk(ntrkid) = set(trk(ntrkid), 1, 'loading');
if currclid == trkinfо(ntrkid, 7)
trkinfо(ntrkid, 7) = 0;
trkinfо(ntrkid, 8) = 0;
end
end
pdc(currclid) = cd;
12. [nextarrival.m] is a MATLAB function which simulates generation of demand.

```matlab
function [packarrevent,packid,ev] =
nextarrival(packarrevent,packid,mrndn,ev,JJD2,tnow,m,routes)

% packarrevent = an array having all the informaiton about packages
% packid = id of next package
% mrndn = an array having mean of distribution to generate demand for each pair of DCs
% ev= an object to maintain event calendar for simulation
% JJJD2 = JJJD matrix for a PLN
% tnow = current time of the simulation
% m = row id for the current package
% routes = cell array having shortest route between each pair of DCs

%get the source DC and destination DC
custdc = packarrevent(m,2);
suppd = packarrevent(m,3);

if m == size(packarrevent,1)  
    packarrevent = packarrevent(1:m-1,:);
else
    packarrevent = packarrevent(2:end,:);
end
packarrevent = [packarrevent(1:m-1,:);packarrevent(m+1:end,:) ];

%get the next package id
packid = packid + 1;

%get the time for next demand arival
narr = genrandnum(mrndn(custdc,suppd),’exp’);
nextarrivaltime = tnow+narr;

%Scheudle the arrival of next demand
ev = putnextevent(ev,[tnow+narrt packid 3]);

%put the information regarding the package scheduled to be generated in
%a seperate matrix
node = length(routes{suppd,custdc});
packarrevent = [packarrevent;packid custdc suppdc 0 node 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 %T[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
36 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
45
46
```matlab
w3 = zeros(dim,dim);
for i=1:dim
  for j=1:dim
    b=rte{i,j};
    if i == j
      b = [b b];
    end
  end
%Considers all the possible paths from P(pred2path)
  for k=1:length(b)-1
    w3(b(k),b(k+1)) = w3(b(k),b(k+1)) + prox1{l}(i,j);
  end
  end
  prox{l,q} = w3;
end
W = w3;
m = 0;
for y = 1:length(p)
  for z = 1:length(TC)
    m = m + 1;
    n = 0;
    for s = 1:length(ul)
      n = n + 1;
      %get the number of trucks
     NT = genAvgTransTime(TC(z),prox{y,s},sum(sum(tp)),IJ,T,l);
      end
      end
      n = n + 1;

%get the truck distribution for different DCs based on
%the demand
    TSI = wtselect(dem/100,1,ceil(sum(sum(NT))));
end
end
```
14. [simulation.m] is a MATLAB function which formats the input before starting the simulation.

1 function [AT,WT] = simulation(nopack,trkcap,tksp,lu,pt,wup,runlen,s,IJD2,xy,Pop,Lf,pick,tp,TSI,W,trkamt,cost,w,B)
2   %
3   %[AT,WT] = simulation(nopack,trkcap,tksp,lu,pt,wup,runlen,s,IJD2,xy,Pop,Lf,pick,tp,TSI,W,trkamt,cost,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   %tksp = truck speed
9   %lu = loading/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 packages is picked
19  %tp = total package demand 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 = tksp;
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 different DCs
43  PK.tp = tp;
44  %packages for each route
45  PK.w = W;
46  PK.ar = w;
47
48  DC.XY = xy;
49  DC.Pop = pop;
50
51  %loading/unloading time at DC
52  DC.lut = lu;
53  %Proximity factor
54  DC.prf = pf;
55  %Process time for demand
56  DC.ppt= pt;
57
58  %Simulation warmup period
59  SIM.wp = wup;
60  %Simulation run length
61  SIM.rl = runlen;
62  %Seed in simulation
63  SIM.seed = s;
64
65  [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.

```matlab
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
npack = 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
% 4 for maximum value and top n
% packages in terms of values, 5 for
% selecting top n bids from all queues using
% 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(npack)
    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] =
53 pln(DC.Pop, DC.XY, IJD2, trkcap(j), trkspeed(k), pf(n), lut(m), nopcode(i), lf(q), nt);
54 %For simulation
55 [ATT, WT] =
56 simulation(nopcode(i), trkcap(j), trkspeed(k), lut(m), pf(n), pt, warmupperiod, runlength, seed(p), IJ
57 D2, DC.XY, DC.Pop, lf(q), qpick(qp), tp, TSI, W, trkamt, cost, w, BV {1, i});
58 end
59 end
60 end
61 end
62 end
16. [startsimulation.m] is a MATLAB function which is the main simulation function.
   function [ATT, avgwt] = startsimulation(DC, TK, PK, SIM, IJD2, pick, cost, B)
   %
   % [ATT, avgwt] = startsimulation(DC, TK, PK, SIM, IJD2, pick)
   % ATT = Average transportation time from simulation
   % avgwt = Average waiting time from simulation
   % DC = Information about DCs in a PLN
   % TK = Information about trucks in a PLN
   % PK = Information about packages in a PLN
   % SIM = Information about simulation
   % IJD2 = IJD matrix for a PLN
   % pick = way a queue of packages is picked - (not used in this simulation)
   % cost = transportation cost and labor cost
   % B = vector having package values
   %
   % Setting the seed for the simulation run
   rand('state', 9501293);
   permseedval = SIM.seed;
   A = full(list2adj(IJD2));
   %
   % initialize global variables
   [ts, lut, ppt, out, td, packid, tp, TSI, W] = initializevariables(TK, DC, PK);
   %
   % Initialize random numbers
   [mrdn] = initializeRandomNumbers(tp);
   %
   global trkinfo;
   global trk;
   global pdc;
   global tnow;
   global routes;
   global allcosts;
   global trkbidtime;
   global trkbidamt;
   global trkbidnc;
   global trkbidloadtime;
   global totpackcost;
   global packtrk;
   totpackcost = 0;
   packtrk = [];
   allcosts = cost;
   %
   % initialize simulation
   [ev, trk, pdc, packarrevent, allpackids, packid, T, routes, trkinfo] = initializeSimulation(tp, IJD2, ts, packid, mrdn, TSI, SIM, TK, pick, W, PK, B);
   %
   trkbidtime = zeros(length(trk), size(T, 2));
   trkbidamt = zeros(length(trk), size(T, 2));
   trkbidnc = zeros(length(trk), size(T, 2));
   trkbidloadtime = zeros(length(trk), size(T, 2));
%For statistics collection and plotting graphs
si = size(tp);
intArrMean = zeros(si);
intArrVar = zeros(si);
intArrNum = zeros(si);
intArrOld = zeros(si);
loadMean = zeros(si);
loadVar = zeros(si);
loadNum = zeros(si);
waitMean = zeros(si);
waitVar = zeros(si);
waitNum = zeros(si);
endsim = 0;
count = 0;
totpack = 0;
toval = 0;
totval2 = 0;
told = 0;
global totpackval;
totpackval = 0;
packsto = {};

avgtrklrd = [0 0];
avgtt = [0 0];
avgdlclrd = [0 0];
avgwtp = [0 0];
avgwt = 0;
td = 0;
ATT = 0;
totearn = 0;
tcount = 0;
toval = 0;
ve = [];
onedc = 0;
onedc = 0;
% 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);

%truck arriving at a DC
if r(3) == 1
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),’cdn’') == 0
%for trucks coming from other DC
currdcid = get(trk(tr),’cdn’');
trk(tr) = set(trk(tr),currdcid,’cdn’');
trk(tr) = set(trk(tr),0,’cdn’');
else
%for the trucks coming straight to the DC. This happens once, at the beginning of the simulation.
currdcid = get(trk(tr),’cdn’');
trk(tr) = set(trk(tr),0,’cdn’');
end

truckreached = tr;

%unload the packages
unloadpackages(tr,currdcid,tnew,routes,A);

%set the truck in the queue at a DC
pdc(currdcid) = set(pdc(currdcid),tr,’truckid’);
truckids = get(pdc(currdcid),’truckid’);

for i = 1:length(truckids)
if get(trk(truckids(i)),’loading’’) == 0
dcont = (get(trk(truckids(i)),’dcont’’));
if length(dcont) > 0
dcont = dcont(1);
if currdcid == dcont
packsto = get(trk(truckids(i)),’pkgcont’’);
packstoload = packsto{1,1};
[ev] = loadpackages(currcid, ev, tnow, lut, ve, routes, W, waitMean, waitVar, waitNum, A, truckids(i), package load); else if dcont == 0 nextdc = routes{currcid, dcont}(2); [ev] = loadpackages(currcid, ev, tnow, lut, ve, routes, W, waitMean, waitVar, waitNum, A, truckids(i), nextdc, 1); end end end
if abs(tnow - 0.1) < 0.00001
%Bidding Process for the package and change in the configuration of the system.
nstep{1,1} = currcid;
nstep{1,2} = [1:length(get(pdc(currcid), 'connecteddc's'))];
nstep{1,3} = tnow;
steo{1,4} = [];
furdc = [];
infopkg = get(pdc(currcid), 'info');

%Bidding at the DC for load formation
[furdc, tr, out, nstep] = bidding(nstep{1,1}, routes, T, tnow, 1, nstep);
out = {};
if size(tr, 1) > 0 trucksel = tr(tr);
trkp = get(trucksel, 'pkgcont');
if get(trucksel, 'loading') == 0 cudc = get(trucksel, 'cdn');
firstdcont = get(trucksel, 'dcont');
if cudc == firstdcont(1)
packsto = get(trucksel, 'pkgcont');
packstoload = packsto{1};

%load the packages on the truck which are at DC where it has reservation
[ev] = loadpackages(cudc, ev, tnow, lut, ve, routes, W, waitMean, waitVar, waitNum, A, tr, packstoload);
end end end

%truck leaving a DC
elseif r(3) == 0
nstep = {};
tr = r(2);
trk(tr) = set(trk(tr), 0, 'loading');

%set the final property of the truck before its departure and
%put the next event of the arrival of the truck, at the next DC, in the event calendar

cdc = get(trk(tr),’cdcn’);
ndc = get(trk(tr),’ndcn’);
transtime = T(cdc,ndc);
pkgtrans = get(trk(tr),’pkgnotnext’);
pdc(ndc) = set(pdc(ndc),[tr tnow+transtime,’incomingtrucks’); cdc = get(pdc(ndc),’connecteddc’);
if size(pkgtrans,1) > 0
nspk = cell(1,length(cdc));

%change the properties and bid amount for the packages whose final
%destination is not next DC
for k=1:size(pkgtrans,1)
routes {pkgtrans(k,3),pkgtrans(k,2)}; ncid = routes {pkgtrans(k,3),pkgtrans(k,2)}(pkgtrans(k,4)+1);
nd = cdc;
pkgtrans(k,9) = pkgtrans(k,10)*A(nd,nid)/dijk(A,pkgtrans(k,3),pkgtrans(k,2));
nd(nd ~~ ncid) = 0;
ndin = find(nd);
pdc(ndc) = set(pdc(ndc),[pkgtrans(k,1) pkgtrans(k,9) tnow+transtime],ndin,’infot’);
pdc(ndc) = set(pdc(ndc),[pkgtrans(k,1) 0],’packtrk’);
nspk{1,nid} = [nspk{1,nid};pkgtrans(k,1)];
end
pkgtrans(:,11) = tnow + transtime;
trk(tr) = set(trk(tr),pkgtrans,’pkgnotnext’);
end
nspt = {};
nspt{1,1} = ndc;
nspt{1,2} = 1:length(cdc);
nspt{1,3} = [];
nspt{1,4} = [];
trk(tr) = set(trk(tr),0,’cdcn’);
trk(tr) = set(trk(tr),cdc,’ldcn’);
truckleaving = tr;
ev = putnextevent(ev,[tnow+transtime,tr,1]);

%Bidding Process and change in the configuration of the system becaue of new load
formation.
outl = {};
bidoc = 0;

while size(nsp,1) > 0
nspn = nsp{1,};
if size(nsp,1) > 1
nsp = nsp{2:end,};
else
nstep = \{\};
end
[furdet,trid,out,nstept] = bidding(nstepn\{1,1\},routes,T,tnow,1,nstepn);
nstep = [nstep;nstept];
if size(trid,1) > 0
furdc = [furdc furdet];
trucksel = trk(trid);
trkp = get(trucksel,'pkgcont');
if get(trucksel,'loading') == 0
cudc = get(trucksel,'cdn');
firstdcont = get((trucksel,'dcont');
if cudc == firstdcont(1)
packsto = get(trucksel,'pkgcont');
packstoload = packsto\{1,1\};
[ev] = loadpackages(cudc,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNumb,A,tru,packstoload);
end
evil
end
end
end

%package demand arriving
elseif r(3) == 3
ev = putnextevent(ev,[tnow,r(2),4]);
%processed package arriving
elseif r(3) == 4
nstep = \{\};
%get the package id
pid = r(2);
allpack = packarrevent(:,1);
allpack(allpack ~= pid) = 0;
m = find(allpack);
%get the source DC and destination DC
currdc = packarrevent(m,3);
destdc = packarrevent(m,2);
infopkg = get(pdc(currdc),'infot');
if currdc == destdc
%to satisfy local demand
rd = rd + 1;
tdlat = 0;
wtlat = 0;
tdold = td;
wtold = avgwt;
valold = totpackval;
curppackval = genrandnum(0,'unif',1);
td = (tdold*rd + tdlat*curppackval)/(valold+curppackval);
avgwt = (wtold*rd + wtlat*curppackval)/(valold+curppackval);
totpackval = totpackval + curppackval;
else
packarrevent(m,4) = l;
packarrevent(m,6) = r(1);
end

packarrevent(m,7) = 0;
packarrevent(m,8) = r(1);
genran = genrandnum(0,'unif',1);
packarrevent(m,10) = genran*dijkstra(A,currdc,destdc);
%Find the next DC for the package
n = routes {currdc,destdc}(2);
dcs = get(pdc(currdc),'connecteddcs');
dest = dc;
dest(dest == n) = 0;
j = find(dest);
packarrevent(m,9) = 
packarrevent(m,10)*A(currdc,ndc)/dijkstra(A,packarrevent(m,3),packarrevent(m,2));
nst{1,1} = currdc;
nst{1,2} = j;
nst{1,3} = tnow;
nst{1,4} = pid;
%Update DC information
pdc(currdc) = set(pdc(currdc),packarrevent(m,:),'data');
pdc(currdc) = set(pdc(currdc),get(pdc(currdc),'currumpack')+1,'currumpack');
pdc(currdc) = set(pdc(currdc),j,j,'numpacksuppby');
pdc(currdc) = set(pdc(currdc),packarrevent(m,9),j,j,'packsuppbay');
pdc(currdc) = set(pdc(currdc),[packarrevent(m,1) packarrevent(m,9) tnow],j,'infot');
packagearrived = pid;
pdc(currdc) = set(pdc(currdc),[pid 0],'packtrk');
%Bidding Process for the package and change in the configuration of the system.
furdc = [];
[furdc,tr,out,nstep] = bidding(nstep{1,1},routes,T,tnow,1,nstep);
furdc = unique(furdc);
out = {};
if size(tr,1) > 0
trksel = trk(tr);
trkp = get(trksel,'pkgcont');
if get(trksel,'loading') == 0
cude = get(trksel,'edcnt');
firstdcont = get(trksel,'dcont');
if cude == firstdcont(1)
packsto = get(trksel,'pkgcont');
packstoload = packsto{1};
[ev] = loadpackages(cude,ev,tnow,lut,ve,routes,W,waitMean,waitVar,waitNum,A,tr,packstoload);
end
end
end
end
while size(nstep,1) > 0
nstep = nstep(1,:);
if size(nstep,1) > 1
nstep = nstep([2:end,:]);
else
  nstepn = {};
end
[furdet, trid, out, nstept] = bidding(nstep{1,1}, routes, T, tnow, 1, nstep);
if size(trid, 1) > 0
  trucksel = trk(trid);
  trkp = get(trucksel, 'pkgcont');
  if get(trucksel, 'loading') == 0
    cudc = get(trucksel, 'cdcn');
    firstdcont = get((trucksel, 'dcont'));
    if cudc == firstdcont(1)
      packsto = get((trucksel, 'pkgcont'));
      packstoload = packsto{1};
    else
      [ev] = loadpackages(cudc, ev, tnow, lut, ve, routes, W, waitMean, waitVar, waitNum, A, trid, packstoload);
    end
  end
end
end
end
end
end
end
end
[packarrevent, packid, ev] = nextarrival(packarrevent, pid, packid, mrndn{currdc, destdc}, ev, JJD2, tnow, m, routes);
else r(3) = 5  % draw the graph
  avgload = 0;
  avgdclad = 0;
  avgwttme = 0;
  tfl = 0;
  for j = 1:length(trk)
    avgload = avgload + get(trk(j), 'currnumpack');
  end
  avgload = avgload/length(trk);
  for km = 1:length(pdc)
    avgdclad = avgdclad + get(pdc(km), 'currnumpack');
  end
  avgdclad = avgdclad/length(pdc);
  tnow = r(1)
  avgwttme = avgwt
  ATT = td
  ATL = avgload
  ADL = avgdclad
  ev = putnextevent(ev, [r(1)+1 5 5]);
  for i = 1:length(pdc)
    pdata = get(pdc(i), 'data');
    if size(pdata, 1) > 0
      for j = 1:size(pdata, 1)
        tdlat = tnow;
        wtlat = tnow;
        tdold = td;
        wtold = avgwt;
      end
    end
  end
end
valold = totpackval;
curppackval = pdata(j,10);

td = (tdold*valold + tdlat*curppackval)/(valold+curppackval);
avgwt = (wtold*valold + wtlat*curppackval)/(valold+curppackval);
totpackval = valold+curppackval;
end
end

for i = 1:length(trk)
pdata = get(trk(i),'data');
if size(pdata,1) > 0
    for j = 1:size(pdata,1)
        tdlat = tnow;
        wtlat = tnow;
        tdold = td;
        wtold = avgwt;
        valold = totpackval;
        curppackval = pdata(j,10);
        td = (tdold*valold + tdlat*curppackval)/(valold+curppackval);
        avgwt = (wtold*valold + wtlat*curppackval)/(valold+curppackval);
        totpackval = valold+curppackval;
    end
end

fid3 = fopen('DataCollection','w+');
fprintf(fid3,'%6.4f\t',td);
fprintf(fid3,'%6.4f\t',avgwt);
fclose(fid3);

endsim = 1;
break;

%Warm up period ends
elseif r(3) == -2
td = 0;
rd = 0;
avgwt = 0;
end
if endsim == 1
break;
end
end

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

    function unloadpackages(tr,currdcid,tnow,routes,A)
    %
    %unloadpackages(tr,currdcid,tnow,routes,A)
    %tr = truck id
    %currdcid = current DC id
    %tnow = current simulation time
    %routes = cell array having routes for each pair of DCs
    %A = matrix having distance between two directly connected DCs
    global trk;
    global pdc;
    global totpackval;
    global totpackcost;
    global trkinfo;

    mrpkg = get(trk(tr),’pkgnotnext’);
    mrpkg = get(trk(tr),’pkgnext’);
    trk(tr) = set(trk(tr),[],’pkgnotnext’);
    trk(tr) = set(trk(tr),[],’pkgnext’);
    if size(mrpkg,1) > 0
        for k = 1:size(mrpkg,1)
            tdlat = tnow - mrpkg(k,6);
            wtlat = mrpkg(k,7);
            dis = dijk(A,mrpkg(k,3),mrpkg(k,2));
            bidamt = mrpkg(k,10)/dis;
            tdtld = td;
            wtold = avgwt;
            valold = totpackval;
            currval = mrpkg(k,10);
            totpackcost = totpackcost + tdlat*currval;
            td = (tdold*valold + tdlat*currval)/(valold+currval);
            avgwt = (wtold*valold + wtlat*currval)/(valold+currval);
            totpackval = valold+currval;
        end
    end
    %If this is the intermediate DC
    if size(mrpkg,1) > 0
        mrpkg(:,8) = tnow;
        mrpkg(:,11) = 0;
        %set the pakcages in the DC
        pdc(currdcid) = set(pdc(currdcid),mrpkg,’data’);
        %set the variables for DC
        pdc(currdcid) =
            set(pdc(currdcid),get(pdc(currdcid),’currumpack’)+size(mrpkg,1),’currumpack’);
        dcs = get(pdc(currdcid),’connectedcs’);
        %increace the number of packages in the respective bays where packages are stored to be
        %transported to their next DCs
        for i = 1:size(mrpkg,1)
            ndc = routes{mrpkg(i,3),mrpkg(i,2)}(mrpkg(i,4)+1);
            dcst = dcs;
48  dest(dest ~= ndc) = 0;
49  j = find(dest);
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,'curnumpack');
18. [updateTrucks.m] is a MATLAB function which updates trucks and DC information after packages are loaded on a truck.

```matlab
function unloadpackages(tr,currdcid,tnow,routes,A)
    %
    %unloadpackages(tr,currdcid,tnow,routes,A)
    %tr = truck id
    %currdcid = current DC id
    %tnow = current simulation time
    %routes = cell array having routes for each pair of DCs
    %A = matrix having distance between two directly connected DCs
    global trk;
    global pdc;
    global totpackval;
    global totpackcost;
    global trkinfo;
    
    mnrpkg = get(trk(tr),'pkgnotnext');
    mrpkg = get(trk(tr),'pkgnext');
    trk(tr) = set(trk(tr),[],'pkgnotnext');
    trk(tr) = set(trk(tr),[],'pkgnext');
    if size(mrpkg,1) > 0
        for k = 1:size(mrpkg,1)
            tdlat = tnow - mrpkg(k,6);
            wtlat = mrpkg(k,7);
            dis = dijk(A,mrpkg(k,3),mrpkg(k,2));
            bidamt = mrpkg(k,10)/dis;
            tdold = td;
            wtold = avgwt;
            valold = totpackval;
            currpackval = mrpkg(k,10);
            totpackcost = totpackcost + tdlat*currpackval;
            td = (tdold*valold + tdlat*currpackval)/(valold+currpackval);
            avgwt = (wtold*valold + wtlat*currpackval)/(valold+currpackval);
            totpackval = valold+currpackval;
        end
    end
end
```

%If this is the intermediate DC
if size(mnrpkg,1) > 0
    mnrpkg(:,8) = tnow;
    mnrpkg(:,11) = 0;
%set the packages in the DC
    pdc(currdcid) = set(pdc(currdcid),mnrpkg,'data');
%set the variables for DC
    pdc(currdcid) = set(pdc(currdcid),get(pdc(currdcid),'currnumpack')+size(mnrpkg,1),'currnumpack');
    dcs = get(pdc(currdcid),'connecteddc');
%increase the number of packages in the respective bays where packages are stored to be transported to their next DCs
    for i = 1:size(mnrpkg,1)
        ndc = routes {mnrpkg(i,3),mnrpkg(i,2)}(mnrpkg(i,4)+1);
```
47  dst = dcs;
48  dst(dst ~= ndc) = 0;
49  j = find(dst);
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.

```matlab
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.

```matlab
function [furdc,tr,done,nstepn,finind,cvimp] = changeConfig(outfin,tnow,T)

% [furdc,tr,done,nstepn,finind,cvimp] = changeConfig(outfin,tnow,T)
% furdc = list of DCs where load formation has changed
% tr = truck id that has accepted the new load at various DCs
% done = "1" if truck has accepted new reservations at DCs
% nstepn = information about the DCs where load formation has changed
% finind = not used now
% cvimp = if value is ",.", the iteration stops
% outfin = information about the various load formations at various DCs
% tnow = current simulation time
% T = matrix having travelling time between directly connected DCs.

global trkinf;
global trk;
global pdc;
global alldc;
global trkbidtime;
global trkbidamt;
global trkbidndc;
global trkbidloadtime;
global packtrk;

cvimp = 0;
done = 0;
lastrow = outfin(size(outfin),:);
currdcid = lastrow(3);
currdcindex = lastrow(2);
currdctime = lastrow(10);
trt = lastrow(1);
timefirst = -1;
firsttimemon = 0;

if length(lastrow{11}) <= 0
    timefirst = lastrow{11};
    firsttimemon = trkinf(trt,3) - trkinf(trt,10);
end
if abs(tnow - trkbidtime(trt,currdcid)) < 0.000001 & trkinf(trt,2) == trkbidndc(trt,currdcid)
    timefirst = trkbidloadtime(trt,currdcid);
    firsttimemon = trkbidamt(trt,currdcid);
end
final = {};
```
finalc = [];
finaltm = [];
finaltm = [];
nstepn = {};
trkinfoi = [];
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};
trkinfoi = trkinfo(tr,:);
trkpkj = get(trk(tr),'pkgcont');
trkdcj = get(trk(tr),'dccont');
pkst1234 = out{size(out,1),4};
96  tcount = 0;
97  trcold= 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  outwobid = outtwo{1,8};
105  [a,b] = sort(outwobid);
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          outwoone = outtwo{1,9};
130      end
131      if out{si,2} == outwoone(2)
132          out{si,9} = outwoone(1);
133  end
134  end
135  totlatbid = 0;
136  flg = 0;
137  totbidamt = 0;
138  totoldbid = 0;
139  pkgcontnew = {};
140  pkgcontnewtwo = {};
141  pkgcontold = {};
142  dccontnew = [];
143  dccontnewtemp = [];
144  dccontold = [];
145  indextndcnvnew = [];
146  indextndcnd = [];
147  indextndcndold = [];
148
134

135
inbidamt = [];
inbidamtold = [];
inbidamtnew = [];
index = 0;
index2 = 0;
count = 0;
dc = [];
depkg = [];
oldpacks = [];
dcindex = [];
 pkgcont = get(ct,'pkgcont');
dcont = get(ct,'dcont');
indexondc = get(ct,'indexonextdc');
inbidamt = get(ct,'bids');
mult = 0;
for i=1:size(out,1)
  if out{i,1} == 0
    if length(out{i,9}) == 0
      count = count + 1;
      pkgcontnew{1,count} = out{i,4};
      %list of new DCs
      dcontnew = [dcontnew out{i,3}];
      %list of new bay numbers
      indexondcnew = [indexondcnew out{i,2}];
      totbidamt = totbidamt + out{i,8};
      %list of amount recieved by truck from various DCs
      inbidamtnew = [inbidamtnew out{i,8}];
    %If new bay number is same as old bay number for the truck
    else
      inbidamtnew = inbidamt;
      mult = 1;
      dc = out{i,3};
      dindex = out{i,2};
      dcontin = out{i,9};
      packs = get(trk(trk),'pkgcont');
      pack = [];
      packbidid = [];
      pack = packs{1,dcontin};
      packid = [];
      if size(pack,1) > 0
        packid = pack(:,1);
      end
      packbid = out{i,4};
      if size(packbid,1) > 0
packbidid = packbid(:,1);
end
[diff,in] = setdiff(packbidid,packid);
dcinfo = get(pdc(dccont(dccontin)),'info');
dcinfof = dcinfo{1,out{i,2}};
dcinfo = get(pdc(dccont(dccontin)),'info');
if length(in) > 0
newload = packbid(in,:);
dcinfot = dcinfo{1,out{i,2}};
if size(dcinfot,1) > 0
if size(newload,1) > 0
dcinfott = dcinfot(:,1);
mem,inct = setdiff(dcinfott,newload(:,1));
dcinfottt = [];
if length(inct) > 0
dcinfottt = dcinfot(inct,:);
end
depkg = dcinfott;
end
end
else
depkg = dcinfo{1,out{i,2}};
end
[diff,in] = setdiff(packid,packbidid);
totoldbid = 0;
oldpacks = [];
if length(in) > 0
depkg = [depkg;pack(in,:)];
totoldbid = totoldbid + sum(pack(in,2));
oldpacks = [oldpacks;pack(in,:)];
end
mem,in] = ismember(packid,packbidid);
inf = find(in);
in = in(inf);
if length(in) > 0
totoldbid = totoldbid + sum(packbid(in,2));
end
packs{1,dccontin} = out{i,4};
pkgcontnew = packs;
transswer = get(trk(tr),'pkgcont');
PKS = out{i,4};
if size(PKS,1) > 0
totlatbid = trkinfo(tr,9) + sum(PKS(:,2)) - totoldbid;
end
depkgc{1,1} = depkg;
if currdcid == dccont(end)
trkinfo = trkinfo(tr,1);
if now - trkinfo > 0.00001
trkinfo = now;
end
trdc = trkinff + dijk(T,trkinfo(tr,2),currdcid);
if currdcid == dcont(end)
trdc = trkinfo(tr,8);
end
if ~abs(trdc - outtwo{1,10}) < 0.00001
flg = 1;
totlbid = 0;
inbidamtnew = [];
[val,index2] = ismember(out{i,3},dcont);

%list of new DCs
dcontnew = dcont(1:index2);

%list of old DCs
dcontold = dcont(index2:end);

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

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

%inbidamtnew - amount truck receives 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};
end
end
end
end
pkgcontnew = pkgcontnewtwo;
totlatbid = totlatbid - totlbid;
end
end
%If truck already has reservations with some of the intermediate DCs
else
dcc = out{i,3};
[val,index] = ismember(dcc,dcont);
end
flag = 0;

% replace packages for the overlapping DC between new and old bids
if mult ~= 1
if length(dcont) > 0
if dcontnew(1) == dcont(end)
if indexdcontnew(1) == indexdcont(end)
pack = [];
packbidid = [];
pack = pkgcont{1,end};
packid = [];
if size(pack,1) > 0
packid = pack(:,1);
end
packbid = pkgcontnew{1,1};
if size(packbid,1) > 0
packbidid = packbid(:,1);
end
oldpacks = [];
[diff,in] = setdiff(packid,packbidid);
if length(in) > 0
oldpacks = pack(in,:);
end
flag = 1;
end
end
end
end
if mult == 0
if index == 0
pkgcontold = pkgcont;
if flag == 1
pkgcontold{1,end} = oldpacks;
end
dcontold = dcont;
indexdcontold = indexdcont;
totoldbid = 0;
if size(pkgcontold,2) > 0
for i = 1:size(pkgcontold,2)
if flag == 1 & i == size(pkgcontold,2)
pk = pkgcont{1,end};
else
pk = pkgcontold{1,i};
end
if size(pk,1) > 0
totoldbid = totoldbid + sum(pk(:,2));
end
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)
chk = pkgcont{1,end};
else
chk = pkgcontold{1,i-index};
end
if size(chk,1) > 0
totoldbid = totoldbid + sum(chk(:,2));
end
end
if index <= length(dcont)
dcontold = dcont(index+1:end);
indexdcontold = indexdcont(index+1:end);
end
dcontnew = [dcont(1:index) dcontnew];
indexdcontnew = [indexdcont(1:index) indexdcontnew];
inbidamtnew = [inbidamt(1:index) inbidamtnew];
for i = 1:size(pkgcontnew,2)
pkgcontnewtwo{1,index+i} = pkgcontnew{1,i};
end
pkgcontnew = pkgcontnewtwo;
end
dcontnew;
totlatbid = trkinfo(tr,9) + totbidamt - totoldbid;
totval = totnumbid-s+1;

%setting the entry information in a cell array
finalout{indval,1} = dcontnew;
finalout{indval,2} = dcontold;
finalout{indval,3} = indexdcontoldnew;
finalout{indval,4} = indexdcontold;
finalout{indval,5} = pkgcontnew;
finalout{indval,6} = pkgcontold;
finalout{indval,7} = inbidamtnew;
finalout{indval,8} = [];

139
396  finalbidouttwo(indval,1) = totlatbid;
397  finalbidout(indval,1) = sum(inbidamtnew);
398  else
399    indval = totnumbid-s+1;
400  if flg == 0
401    %putting the entrie information in a cell array
402    finalout{indval,1} = dcont;
403    finalout{indval,2} = dcindex;
404    finalout{indval,3} = [];
405    finalout{indval,4} = dc;
406    finalout{indval,5} = pkgcontnew;
407    finalout{indval,6} = dcpkg;
408    finalout{indval,7} = inbidamtnew;
409    finalout{indval,8} = [];
410  finalbidouttwo(indval,1) = totlatbid;
411  finalbidout(indval,1) = sum(inbidamtnew);
412  elseif flg == 1
413    %putting the entrie information in a cell array
414    finalout{indval,1} = dcontnew;
415    finalout{indval,2} = dcontold;
416    finalout{indval,3} = indexondcnew;
417    finalout{indval,4} = indexondcold;
418    finalout{indval,5} = pkgcontnew;
419    finalout{indval,6} = pkgcontold;
420    finalout{indval,7} = inbidamtnew;
421    finalout{indval,8} = index2;
422  finalbidouttwo(indval,1) = totlatbid;
423  finalbidout(indval,1) = sum(inbidamtnew);
424  end
425  end
426  tc = 0;
427
428  %calculate the transporation cost for the entire stretch of journey
429  if length(finalout{indval,3}) > 0
430    indexndtc = finalout{indval,3};
431    dcontc = finalout{indval,1};
432    ldc = dcontc(end);
433    ldetc = indexndtc(end);
434    contedcc = getpdc(ldc,'connecteddc');
435    llldc = conteddc(ldetc);
436    tc = (dijkstra(T*60,dcontc(1),ldc)+dijkstra(T*60,ldc,llldc))*allcosts(1);
437  else
438    tc = trkinfoc(tr,10);
439  end
440  finalbidoutcost(indval,1) = finalbidout(indval,1) - tc;
441  finaloutcost(indval,1) = tc;
442  end
443
444  %get the series of DCs and bay numbers which gives maximum amount of money
445  %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(currvecid,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 ldeo = 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 = indexndctdc;
478 end
479 deconttc = finalout{bidind,1};
480 ldc = deconttc(end);
481 ldctcin = indexndctc(end);
482 conteddc = get(pdc(ldc),'connecteddc');
483 lldc = conteddc(ldctcin);
484 tclast = dijk(T*60,ldc,lldc)*2;
485 end
486 if tclast == 0
487 tclast = 9999999999;
488 end
489 if done ~= 2
490
491 %Check if net amount truck gets is more than the current amount it gets
492 if cvvlast - tclast > 0.00001
493 if length(decont) > 0
494 decontnews = finalout{bidind,1};
495 if decont(end) == decontnews(end)
if currdcid == decontnews(end)
if cv2 - cv1 > 0.00001
  done = 2;
elseif currdcid == decontnews(end)
  if ~abs(trkinfo(tr,8) - outr{10}) < 0.00001
    cvlast = inbidamtnew(end);
  elseif abs(trkinfo(tr,8) - outr{10}) < 0.00001
    indexndtc = indextondc;
    deconttc = dccont;
    ldc = deconttc(end);
    ldctcin = indexndtc(end);
    conteddc = get(pdc(ldc), connecteddcs);
    llcd = conteddc(ldctcin);
    tlastold = dijk(T*60, ldc, llcd)*2;
    cvlastold = inbidamt(end);
    cvlast = inbidamtnew(end);
    if cv > 0.00001
      if ((cvlast-tlast) - (cvlastold - tlastold)) > 0.00001
        done = 2;
      elseif abs((cvlast-tlast) - (cvlastold - tlastold)) < 0.00001
        if cv2 - cv1 > 0.00001
          done = 2;
        end
      end
    end
  end
else
  if length(finalout{bidind,8}) == 0
    if cv - cvo > 0.00001
      done = 2;
    elseif abs(cv-cvo) < 0.00001
      if cv2 - cv1 > 0.00001
        done = 2;
      end
    end
  else
    index2 = finalout{bidind,8};
    pkg2new = finalout{bidind,5};
    pkg2oldin = pkgcont{1,index2};
    pkg2newin = pkg2new{1,end};
    if cmparr(pkg2oldin,pkg2newin) == 1
      if cv - cvo > 0.00001
        done = 2;
      end
    end
  end
end

142
elseif abs(cv-cvo) < 0.00001
if cv2 - cv1 > 0.00001
done = 2;
end
end
elseif abs(cv-cvo) < 0.00001
if cv2 - cv1 > 0.00001
done = 2;
end
else
index2 = finalout{bidind,8};
pkg2new = finalout{bidind,5};
pkg2oldin = pkgcont{1,index2};
pkg2newin = pkg2new{1,end};
if cmparr(pkg2oldin,pkg2newin) == 1
if cv - cvo > 0.00001
done = 2;
elseif abs(cv-cvo) < 0.00001
if cv2 - cv1 > 0.00001
done = 2;
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
%If truck gets more money than its current amount
if done == 2
if timefirst == -1
%change the configuration of the truck
timefirst = outt{10};
firsttimemon = cv;
final = finalout;
finalc = finaloutcost(bidind);
finaltm = finalbidout(bidind);
finalvm = finalbidouttwo(bidind);
end
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(ou{10} - timefirst) < 0.00001
    final = finalout;
    finalc = finaloutcost(bidind);
    finaltm = finalbidout(bidind);
    finalvm = finalbidouttwo(bidind);
  else
    cvnew = cv - (ou{10} - timefirst)*allcosts(2);
    if cvnew - firsttimemon > 0.00001
      timefirst = ou{10};
      firsttimemon = cv;
      final = finalout;
      finalc = finaloutcost(bidind);
      finaltm = finalbidout(bidind);
      finalvm = finalbidouttwo(bidind);
    end
  end
else
  cvnew = cv - (ou{10} - timefirst)*allcosts(2);
  if cvnew - firsttimemon > 0.00001
    final = finalout;
    finalc = finaloutcost(bidind);
    finaltm = finalbidout(bidind);
    finalvm = finalbidouttwo(bidind);
    timefirst = ou{10};
    firsttimemon = cv;
  end
end
finalout = {};
finaloutcost = [];
finalbidout = [];
finalbidouttwo = [];
bidind = 1;
finalout = final;
finaloutcost = finalc;
finalbidout = finaltm;
finalbidouttwo = finalvm;
if size(final,1) > 0
if length(final{bidind,1}) == 1
    trkinfo(tr,10) = finaloutcost(bidind,1);
    dconbid = finalout{bidind,1};
    dcontr = get(trk(tr),'dcon');
    if length(dcontr) > 0
        if dconbid(end) == dcontr(end)
            trkinfo(tr,7) = out{3};
            trkinfo(tr,8) = timefirst;
            elseif currdcid == dconbid(end)
            trkinfo(tr,7) = out{3};
            trkinfo(tr,8) = timefirst;
        end
    end
    done = 1;
    if length(finalout{bidind,3}) > 0
        dconnew = finalout{bidind,1};
        dconold = finalout{bidind,2};
        indextondcnew = finalout{bidind,3};
        indextondcold = finalout{bidind,4};
        pkgcontnew = finalout{bidind,5};
        pkgcontold = finalout{bidind,6};
        inbidamtnew = finalout{bidind,7};
        otr = 0;
        otr2 = [];
        for i = 1:length(dconnew)
            dcinfo = get(pdc(dconnew(i)),'info');
            dcinfo = dcinfo(1,indextondcnew(i));
            pkg = pkgcontnew(1,i);
            get(pdc(dconnew(i)),'info');
            if size(pkg,1) > 0
                pkgtr = get(pdc(dconnew(i)),'packtr');
                pkgtr = pkgtr(:,1);
                [x,y] = ismember(pkg(:,1),pkgtr);
                otr = pkgtr(y,2);
                otr = unique(otr);
                otr2 = [otr2;otr];
                pkgtr(y,2) = tr;
                pkgtr = pkgtr(:,2);
                pkgtr(pkgtr == tr) = 0;
                a = find(pkgtr);
                pkgtr(a,1) = 0;
                [x,y] = setdiff(pkgtrtr,pkg(:,1));
                pkgtr(a,y,2) = 0;
                pdc(dconnew(i)) = set(pdc(dconnew(i)),pkgtr,'setpacktr');
            end
        end
        %change the configuration of new DCs
        if size(dcinfo,1) > 0
[diff,in] = setdiff(dcinfot(:,1),pkg(:,1));
if length(in) > 0
dcinfottwo = dcinfot(in,:);
pdc(dcontnew(i)) = setpdc(dcontnew(i),dcinfottwo,indextondcnew(i),’setinfo’);
else
pdc(dcontnew(i)) = setpdc(dcontnew(i),[],indextondcnew(i),’setinfo’);
end
else
pdc(dcontnew(i)) = setpdc(dcontnew(i),[],indextondcnew(i),’setinfo’);
end
pkgtrk = get(pdc(dcontnew(i)),’packtrk’);
if size(pkgtrk,1) == 0
pkgtrd = pkgtrk(:,1);
pkgtr = pkgtrk(:,2);
pkgtr(pkgtrtrk == tr) = 0;
a = find(pkgtr);
if length(a) == 0
pkgtr(a,2) = 0;
pdc(dcontnew(i)) = setpdc(dcontnew(i),pkgtrk,’setpacktrk’);
end
end
end
end
otr = unique(otr);

%Change the configuration of the truck
ct = set(ct,pkgcontnew,’pkgcont’);
ct = set(ct,dcontnew,’dcont’);
ct = set(ct,indextondcnew,’indextdc’);
ct = set(ct,inbidamtnew,’bids’);

%Change the configuration of the old dc
for j = 1:length(dcontold)
pdc(dcontold(j)) = setpdc(dcontold(j),pkgcontold{1,j},indextdcold(j),’info’);
end
furdc = dcontold;
trk(out {size(out,1),1}) = ct;
else
trk(tr) = set(trk(tr),finalout{bidind,5},’pkgcont’);
dcpkgf = finalout{bidind,6};
pdc(finalout{bidind,4}) =
set(pdc(finalout{bidind,4}),dcpkgf{1,1},finalout{bidind,2},’setinfo’);
trk(out {size(out,1),1}) = set(trk(out {size(out,1),1}),finalout{bidind,7},’bids’);
furdc = finalout{bidind,4};
[x,y] = ismember(currccd,finalout{bidind,1});
pkg = pkg{1,y};
pkgtr = get(pdc(currccd),’packtrk’);
pkgtr = pkgtr(:,1);
[x,y] = ismember(pkg(:,1),pkgtrt);
otr = pkgtr(y,2);
otr = unique(otr);
pkgtr(y,2) = tr;
pkgtrt = pkgtr(:,2);
pkgtrt(pkgtrt == tr) = 0;
a = find(pkgtrt);
pkgtrt = pkgtr(a,1);
[x,y] = setdiff(pkgtrt,pkg(:,1));
if length(y) > 0
pkgtr(a(y),2) = 0;
end
pdc(currdcid) = set(pdc(currdcid),pkgtrt,'setpacktrk');
end
otr1 = length(find(otr));
if otr1 == 0
for i = 1:length(otr)
if otr(i) == tr
if otr(i) == 0
trtk = otr(i);
dccon = get(trtk,'dccon');
pkgdcn = get(trtk,'pkgdcn');
indexon = get(trtk,'indexonextdc');
inbidon = get(trtk,'bids');
[x,y] = ismember(currdcid,dccon);
trtk(trtk) = set(trtk,{},'dccon');
trtk(trtk) = set(trtk,{},'indexonextdc');
trtk(trtk) = set(trtk,{},'bids');
trkinf0(trtk,10) = 0;
trkinf0(trtk,8) = 0;
trkinf0(trtk,7) = 0;
y = 1;
for j = y:length(dccon)
pkg = pkgdcn{1,j};
pkgtr = get(pdc(dccon(j)),'packtrk');
if size(pkgtr,1) > 0
pkgtrt = pkgtr(:,1);
if size(pkg,1) > 0
[x,z] = ismember(pkg(:,1),pkgtrt);
pkgtrn = pkgtrn(z,2);
pkgtrn(pkgtrn == trtk) = 0;
b = find(pkgtrn);
if length(find(b)) == 0
pkgtrn(z(b),2) = 0;
end
pdc(dccon(j)) = set(pdc(dccon(j)),pkgtrt,'setpacktrk');
pkgsunr = sum(pkg(:,2));
pkgsunv = inbidon(j);
trkinf0(trtk,3) = trkinf0(trtk,3) - pkgsunv;
trkinfo(trtk,9) = trkinfo(trtk,9) - pkgsumr;
end

%Arrange information about the DCs which have gone through new load

%formation
nsnep{y+1,1} = decon(j);
nstep{y+1,2} = indexonj(t);
nstep{y+1,3} = []; nstep{y+1,4} = [];
end end end
end

furdc = [furdc;currccid];
if length(furdc) > 0
n_pkg = finalout{bidind,6};
if finalout{bidind,3} > 0
nindex = finalout{bidind,4};
else
nindex = finalout{bidind,2};
end

/tn = size(nstep,1);
for i = 1:length(furdc)
tn = tn + 1;
nstep{tn,1} = furdc(i);
nstep{tn,2} = [];
nstep{tn,3} = [];
nstep{tn,4} = [];
end
end
furdc = [currccid furdc];
trkinfo(trt,3) = finalbidout(bidind);
trkinfo(trt,9) = finalbidouttwo(bidind);
trkbidtime(trt,currccid) = tnow;
trkbidamt(trt,currccid) = firsttimemon;
trkbidndc(trt,currccid) = trkinfo(trt,2);
trkbidloadtime(trt,currccid) = timefirst;
else
tr = [];
done = 0;
if length(dccont) > 0
for i = 1:length(dccont)
pkgc = pkgcont{1,i};
pkgdc = get(pdc(dccont(i)),'info');
pkgdcind = pkgdc{1,indexoncontdc(i)};
if size(pkgdcind,1) > 0
if size(pkgc,1) > 0
pkgcid = pkgc(:,1);

845  pkgdcindid = pkgdcind(:,1);
846  [x,y] = setdiff(pkgdcindid, pkgcid);
847  pde(dcont(i)) = set(pde(dcont(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(dcont) > 0
857     for i = 1:length(dcont)
858     pkgc = pkgcont{1,i};
859     pkgdc = get(pdc(dcont(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     pde(dcont(i)) = set(pde(dcont(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), 'dcont');
875     end