Zuolin Li, 998829351

Hanyuan Dong, 998983129

ECS152A Project1

Utilization:

 \rho  = \frac{\lambda }{\mu }\,\!  

Mean Queue Length (Infinite Buffer size):

E(n) = \frac{{\rho }}{{1 - \rho }} \,\!

Mean Queue Length (Finite Buffer size)(substitute n+1 with N when calculating)

E(n) = \frac{{\left( \rho  \right)}}{{\left( {1 - \rho } \right)^{} }}\frac{{1 - \left( {N + 1} \right)\left( \rho  \right)^N  + N\rho ^{N + 1} }}{{1 - \rho ^{N + 1} }}
\,\!

Buffer size 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 |
| Average queue length | 0.2248 | 0.4913 | 0.6791 | 0.8533 | 0.935 |
| Utilization | 19.89% | 40.07% | 60.5% | 79.9% | 90.83% |
| Package dropped | 1565 | 5404 | 10208 | 14861 | 17613 |
| Theoretic Average queue length | 0.2258 | 0.4615 | 0.6734 | 0.8524 | 0.9299 |
| Theoretic Utilization | 20% | 40% | 60% | 80% | 90% |

Buffer size 20:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 |
| Average queue length | 0.2485 | 0.6555 | 1.4369 | 3.9487 | 6.7836 |
| Utilization | 19.93% | 39.91% | 59.51% | 80.06% | 90.19% |
| Package dropped | 0 | 0 | 0 | 80 | 667 |
| Theoretic Average queue length | 0.2499 | 0.6667 | 1.4999 | 3.8364 | 6.5968 |
| Theoretic Utilization | 20% | 40% | 60% | 80% | 90% |

Buffer size 50:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 |
| Average queue length | 0.254 | 0.6716 | 1.4945 | 3.825 | 8.3711 |
| Utilization | 20.18% | 40.09% | 60.26% | 79.42% | 89.47 |
| Package dropped | 0 | 0 | 0 | 0 | 12 |
| Theoretic Average queue length | 0.25 | 0.6667 | 1.4999 | 3.9994 | 8.7820 |
| Theoretic Utilization | 20% | 40% | 60% | 80% | 90% |

Buffer size Infinite:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.1 | 0.25 | 0.4 | 0.55 | 0.65 | 0.8 | 0.9 |
| Average queue length | 0.1114 | 0.3329 | 0.6841 | 1.1937 | 1.8227 | 4.1233 | 9.2474 |
| Utilization | 10.0% | 24.91% | 40.46% | 54.75% | 64.66% | 80.2% | 90.55% |
| Package dropped | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Theoretic Average queue length | 0.1111 | 0.3333 | 0.6667 | 1.222 | 1.857 | 4.000 | 9.000 |
| Theoretic Utilization | 10% | 25% | 40% | 55% | 65% | 80% | 90% |

Queue-length vs λ Utilization vs λ

Code:

Event.java:

/\*\*

\* Author: Zuolin Li, 998829351

\* Hanyuan Dong, 998983129

\* ECS152A Project1 Phase1

\* Simulation Analysis of a Network Protocol

\*/

**package** ECS152A;// can be delete,eclipse package match;

**public** **class** Event **implements** Comparable<Event>{

**public** **double** eventTime;

**public** **double** serviceTime;

**public** **int** eType;//0 is arrival, 1 is depart;

@Override

//a method to sort a arrayList by its eventTime;

**public** **int** compareTo(Event t){

**double** comparetime=((Event)t).eventTime;

**return** (**int**) ((**this**.eventTime-comparetime)\*1000);

}

}

Phase1.java:

/\*\*

\* Author: Zuolin Li, 998829351

\* Hanyuan Dong, 998983129

\* ECS152A Project1 Phase1

\* Simulation Analysis of a Network Protocol

\*/

**package** ECS152A;// can be delete,eclipse package match.

**import** java.text.DecimalFormat;

**import** java.util.\*;

**public** **class** Phase1 {

**private** **static** **int** *maxBuffer*;

**private** **static** **double** *arrivalR*;

**private** **static** **double** *transR*;

**private** **static** **int** *length* = 0;

**private** **static** **double** *avgLength* = 0;

**private** **static** **double** *ult* = 0;

**private** **static** **int** *packageDropped* = 0;

**private** **static** **double** *time*=0;

**public** **static** **void** main(String[] args) {

//Initialization

**double** busyTime=0;

**double** packageArea=0;

ArrayList<Event> GEL=**new** ArrayList<Event>();

Queue<Event> queue=**new** LinkedList<Event>();

/\*transR=u=1

arrivalR=0.1,0.25,0.4,0.55,0.65,0.8,0.9 with infinite buffer size

arrivalR=0.2,0.4,0.6,0.8,0.9 with maxBuffer=1,20,50

\*/

*arrivalR*=0.9;

*transR*=1.0;

*maxBuffer*=1;

//Integer.MAX\_VALUE;

//create the first arrival event and insert to GEL;

Event event=**new** Event();

event.eventTime=*time*+*negExpDTime*(*arrivalR*);

event.serviceTime=*negExpDTime*(*transR*);

event.eType=0;

GEL.add(event);

**for**(**int** i=0;i<100000;i++){

//get the first event from GEL;

Event evet=GEL.remove(0);

//System.out.println("Time: "+time);

packageArea+=*length*\*(evet.eventTime-*time*);

*time*=evet.eventTime;

//printEvent(evet,length,packageArea);

//if it is an arrival event;

**if**(evet.eType==0){

Event newEvent= **new** Event();

newEvent.eType=0;

newEvent.eventTime=*time*+*negExpDTime*(*arrivalR*);

//System.out.println("event time: "+newEvent.eventTime);

newEvent.serviceTime=*negExpDTime*(*transR*);

GEL.add(newEvent);

Collections.*sort*(GEL);

busyTime=busyTime+newEvent.serviceTime;

//System.out.println(busyTime);

**if**(*length*==0){// empty queue;

Event depart=**new** Event();

depart.eType=1;

depart.eventTime=*time*+newEvent.serviceTime;

GEL.add(depart);

Collections.*sort*(GEL);

*length*++;

}**else** **if**(*maxBuffer*>*length*-1){// the queue is not full;

queue.add(newEvent);

*length*++;

}

**else**{//the queue is full, drop the package;

*packageDropped*++;

}

}

//else must be departure;

**else**{

*length*--;

**if**(*length*>0){ //if there are package in queue;

Event outEvent=queue.remove();

Event depart2=**new** Event();

depart2.eType=1;

depart2.eventTime=*time*+outEvent.serviceTime;

GEL.add(depart2);

Collections.*sort*(GEL);

} //if no package in queue, just continue;

}

//if(GEL.size()==0) System.out.println("all gone");;

}

//out-put statistics;

//System.out.println(busyTime + " "+time);

*ult*=*roundTwoDecimal*(busyTime/*time*);

*avgLength*=*roundTwoDecimal*(packageArea/*time*);

/\*

\*System.out.println("Event number remaining: "+GEL.size());

\*while(GEL.size()!=0) printEvent(GEL.remove(0),length,packageArea);

\*/

System.***out***.println("MAXBUFFER: " + *maxBuffer*);

System.***out***.println("Arrival Rate(λ): " + *arrivalR*);

System.***out***.println("Transmission Rate(u): " + *transR*);

System.***out***.println("Average queue length: " + *avgLength*);

System.***out***.println("Utilization:" + *ult*\*100 +"%");

System.***out***.println("Package dropped: " + *packageDropped*);

//System.out.println(negExpDTime(0.1));

}

/\*\* testing if the order is right.

public static void printEvent(Event evet,int l,double a){

System.out.println("Event out: type: "+evet.eType+" EventTime: "+evet.eventTime

+" ServiceTime: "+evet.serviceTime+" Length: "+l+"Area: "+a);

}\*/

//round double to 4 decimal;

**public** **static** **double** roundTwoDecimal(**double** d){

DecimalFormat temp=**new** DecimalFormat("#.####");

**return** Double.*valueOf*(temp.format(d));

}

**public** **static** **double** negExpDTime(**double** rate){

**double** u= Math.*random*();

**return** ((-1/rate)\*Math.*log*(1-u));

}

}

Extra credit:

Phase1Pareto.java:

/\*\*

\* Author: Zuolin Li, 998829351

\* Hanyuan Dong, 998983129

\* ECS152A Project1 Phase1

\* Simulation Analysis of a Network Protocol

\*/

**package** ECS152A;// can be delete,eclipse package match.

**import** java.text.DecimalFormat;

**import** java.util.\*;

**public** **class** Phase1Pareto {

**private** **static** **int** *maxBuffer*;

**private** **static** **double** *arrivalR*;

**private** **static** **double** *transR*;

**private** **static** **int** *length* = 0;

**private** **static** **double** *avgLength* = 0;

**private** **static** **double** *ult* = 0;

**private** **static** **int** *packageDropped* = 0;

**private** **static** **double** *time*=0;

**public** **static** **void** main(String[] args) {

//Initialization

**double** busyTime=0;

**double** packageArea=0;

ArrayList<Event> GEL=**new** ArrayList<Event>();

Queue<Event> queue=**new** LinkedList<Event>();

/\*transR=u=1

arrivalR=0.1,0.25,0.4,0.55,0.65,0.8,0.9 with infinite buffer size

arrivalR=0.2,0.4,0.6,0.8,0.9 with maxBuffer=1,20,50

\*/

*arrivalR*=0.2;

*transR*=1.0;

*maxBuffer*=1;

//Integer.MAX\_VALUE;

//create the first arrival event and insert to GEL;

Event event=**new** Event();

event.eventTime=*time*+*paretoDTime*(*arrivalR*);

event.serviceTime=*negExpDTime*(*transR*);

event.eType=0;

GEL.add(event);

**for**(**int** i=0;i<100000;i++){

//get the first event from GEL;

Event evet=GEL.remove(0);

packageArea+=*length*\*(evet.eventTime-*time*);

*time*=evet.eventTime;

//if it is an arrival event;

**if**(evet.eType==0){

Event newEvent= **new** Event();

newEvent.eType=0;

newEvent.eventTime=*time*+*paretoDTime*(*arrivalR*);

newEvent.serviceTime=*negExpDTime*(*transR*);

GEL.add(newEvent);

Collections.*sort*(GEL);

**if**(*length*==0){// empty queue;

Event depart=**new** Event();

depart.eType=1;

depart.eventTime=*time*+newEvent.serviceTime;

GEL.add(depart);

Collections.*sort*(GEL);

*length*++;

busyTime=busyTime+newEvent.serviceTime;

}**else** **if**(*maxBuffer*>*length*-1){// the queue is not full;

queue.add(newEvent);

*length*++;

busyTime=busyTime+newEvent.serviceTime;

}

**else**{//the queue is full, drop the package;

*packageDropped*++;

}

}

//else must be departure;

**else**{

*length*--;

**if**(*length*>0){ //if there are package in queue;

Event outEvent=queue.remove();

Event depart2=**new** Event();

depart2.eType=1;

depart2.eventTime=*time*+outEvent.serviceTime;

GEL.add(depart2);

Collections.*sort*(GEL);

} //if no package in queue, just continue;

}

}

//out-put statistics;

*ult*=*roundTwoDecimal*(busyTime/*time*);

*avgLength*=*roundTwoDecimal*(packageArea/*time*);

System.***out***.println("MAXBUFFER: " + *maxBuffer*);

System.***out***.println("Arrival Rate(λ): " + *arrivalR*);

System.***out***.println("Transmission Rate(u): " + *transR*);

System.***out***.println("Average queue length: " + *avgLength*);

System.***out***.println("Utilization:" + *ult*\*100 +"%");

System.***out***.println("Package dropped: " + *packageDropped*);

//System.out.println(negExpDTime(0.1));

}

/\*\* testing if the order is right.

public static void printEvent(Event evet,int l,double a){

System.out.println("Event out: type: "+evet.eType+" EventTime: "+evet.eventTime

+" ServiceTime: "+evet.serviceTime+" Length: "+l+"Area: "+a);

}\*/

//round double to 4 decimal;

**public** **static** **double** roundTwoDecimal(**double** d){

DecimalFormat temp=**new** DecimalFormat("#.####");

**return** Double.*valueOf*(temp.format(d));

}

**public** **static** **double** negExpDTime(**double** rate){

**double** u= Math.*random*();

**return** ((-1/rate)\*Math.*log*(1-u));

}

**public** **static** **double** paretoDTime(**double** rate){

**double** u=Math.*random*();

//System.out.println(u);

**return** u/Math.*pow*(rate, 2) ;

}

}

Extra Credit (Pareto distribution)

Buffer size 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 |
| Average queue length | 0.0827 | 0.3634 | 0.8247 | 1.2684 | 1.4295 |
| Utilization | 7.97% | 30.84% | 60.64% | 81.49% | 87.55% |
| Package dropped | 91 | 1530 | 8447 | 22166 | 30080 |

Buffer size 20:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 |
| Average queue length | 0.083 | 0.3983 | 1.9105 | 18.706 | 19.889 |
| Utilization | 7.96% | 31.97% | 72.41% | 100% | 100% |
| Package dropped | 0 | 0 | 5 | 12618 | 23338 |

Buffer size 50:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 |
| Average queue length | 0.0835 | 0.4035 | 1.8464 | 48.55 | 49.86 |
| Utilization | 8.01% | 32.23% | 71.95% | 100% | 100% |
| Package dropped | 0 | 0 | 0 | 12299 | 23523 |

Buffer size Infinite:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Data / Arrival Rate | 0.1 | 0.25 | 0.4 | 0.55 | 0.65 | 0.8 | 0.9 |
| Average queue length | 0.0203 | 0.1351 | 0.3983 | 1.1259 | 3.5712 | 6158.2 | 11719.7 |
| Utilization | 2.01% | 12.58% | 31.98% | 60.09% | 84.26% | 100% | 100% |
| Package dropped | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Queue-length vs λ Utilization vs λ