Network tracing using WIDE Data Repository to study the characteristics of Internet traffic

Term Project submitted by
Anirban Banerjee,
Piyush Ranjan Satapathy,
Trivikram G Phatak and
Varun Kohli

In partial fulfillment of the requirements for completion of the course,
CS204 (Advanced Computer Networks) instructed by
Dr Mart Molle

Offered by the
Department of Computer Science & Engineering
At
University of California Riverside

Contents

Abstract:
As Internet technology becomes more diverse, and the network becomes more complicated, the characteristics of traffic also become more diverse. It has already become difficult to make projections for the next year from this year's data sets. Therefore, it is needed to develop new methods of measurement and analysis, and a second look must be taken at the importance of measurement and analysis of the Internet. However, this is a field in which patience is required and good results are hard to obtain. Hence, organizational support is needed to back up efforts in this field. We have obtained data from Wide data repository on a daily basis for 1 week and then weekly basis for 4 weeks and then monthly basis for 2 months. We have calculated and plotted the network characterization curves such as cumulative arrival curve, minimal arrival curve and autocorrelation functions. From this we do some analysis to reach at a consensus over some prediction and periodicity over data on a time scale.

1. Introduction:

1.1 Internet: An effective way of communication?

In Internet networking, the tremendous growth in the numbers of hosts, networks, network types, and network peering points results in a general lack of understanding of network performance. It is believed that consistent measurement of well-defined performance metrics would enable better network engineering and operations. As the Internet becomes larger and more complex, sound methodologies to characterize Internet performance and a scalable measurement infrastructure become indispensable.

1.2 An overlook of WIDE Traffic Data Repository Project:

So there comes a need of a database for actual traffic data and having sufficient information to analyze the end-to-end characteristics of Internet or intranet traffic without disclosing information about the end users. After discussing several inherent problems associated with modeling and simulations on Internet or intranet traffic, the required data that must be captured in sufficient amounts will be described as well as the format of the database and capturing program. The first commonly used traffic data were captured on 3 October 1989 at the Bell core Morristown Research and Engineering Facility. The data files were in ASCII format, consisting of one 20-byte line for each arriving Ethernet packet. The next generation of traffic data collections was based on the tcpdump program. Such a traffic data repository is WIDE project. In this repository, Traffic traces are collected by *tcpdump* and, after removing privacy information; the traces are made open to the public.

The WIDE project is a research consortium in Japan established in 1987. The focus of the project is empirical study on a live large-scale Internet. Thus, WIDE runs its own Internet tested carrying both commodity traffic and research experiments and gives us a high quality of internet data between USA and Japan.

1.3 Our Objective:

The goals of this paper are to use the above said data archives and to test how real Internet traffic compares with the various theoretical traffic models we studied in the lecture. With respect to network calculus, we calculated the cumulative arrival curves, minimum arrival curves and autocorrelation functions for various daily trace files from the WIDE data archive. We plotted all the curves and then we analyzed how well behaved they are. We also analyzed how different curves generated from traces collected on different days show a consistent structure or change significantly and whether the changes are unpredictable, or correlated to the days of the week.

2 An Overview of Network Calculus (Theory From Lecture)

2.1 Arrival Curve:

When packets arrive into a queue associated with an output link of a switch, they arrive over an input link and hence arrive at a finite speed. A packet of length L bits arriving over a link of bit rate C will start arriving at some time t and will finish arriving at time (t + L/C). The next packet on the same input link cannot start arriving before (t + L/C). Thus the inter arrival times between packets on a link are bounded below by the transmission times of the packets on the link. Assuming that the link scheduler is non-idling or work conserving we always see the no of bits arrived at any unit time and if these two plotted, we say it to be an arrival curve of a networking system.

Definition:

Given a wide-sense increasing function α defined for $t \geq 0$ we say that a flow R is constrained by α if and only if for all $s \leq t$:

$$R(t) - R(s) \leq \alpha(t-s)$$

We say that R has α as an arrival curve, or also that R is α smooth.

2.2 Cumulative Arrival Curve:

It is always convenient to describe data flows by means of the cumulative function R (t), defined as the number of bits seen on the flow in time interval [0,t]. By convention, we take R (0) = 0, unless otherwise specified. Function R is always wide-sense increasing. We can use a discrete or continuous time model .In real systems, there is always a minimum granularity (bit, word, cell or packet), therefore discrete time with a finite set of values for R(t) could always be assumed. However, it is often computationally simpler to consider continuous time, with a function R that may be continuous or not. If R (t) is a continuous function, we say that we have a fluid model. Otherwise, we take the convention that the function is either right or left-continuous.

Definition:

Suppose we assume that the cumulative arrival curve function, R (t) has a derivative r (t) which is defined as; r (t) = (dR / dt), then r is called the rate function. So the cumulative function R (t) us defined as below;

$$R(t) = \int_0^t r(s)\, ds$$

Where r (s) = Instant Arrival rate at time "s".

2.3 Minimum Arrival Curve:

Considering a given flow R (t), we can only find one minimal arrival curve. Lets assume that R (t) is known from measurements.

Definition:

Considering a flow R (t) such as $R(t)_{t \geq 0}$ the minimum arrival curve is defines as the function (R ∅ R) where ∅ symbolizes the sub-additive deconvolution operator or min-plus deconvolution operator.

$$(R \oslash R)(t) = \sup_{N \geq 0} \{R(t+N) - R(N)\}$$

So (R∅R) (t) is an arrival curve that constraints flow A (t). Also from definition we can get as below;

$$R(t) - R(s) \leq \alpha(t-s)$$

$$\leq \sup_{N \geq 0} \{R(t-s+N) - R(N)\}$$

$$= (R \oslash R)(t-s)$$

Here, R∅R is an arrival curve for R (t) and it is the minimum arrival curve and it is sub-additive as said above.

2.4 Auto Correlation Function:

Let X (t) = Network traffic Measurement (bits or packets) over the t^{th} interval of length Δ. So X (t) is defined as;

$$X(t) = R(t\Delta) - R((t-1)\Delta)$$

Definition:

The autocorrelation function (ACF), $\rho(k)$, is measured as the similarity between a series X (t) and a shifted version of itself, X_{t+k} ;

$$\rho(k) = E[(X_t - \mu)(X_{t+k} - \mu)] \; / \; \sigma^2$$

Where μ = Sample mean of the network traffic measurement over the time series
σ = Standard deviation of the time series

3. Our Implementation:

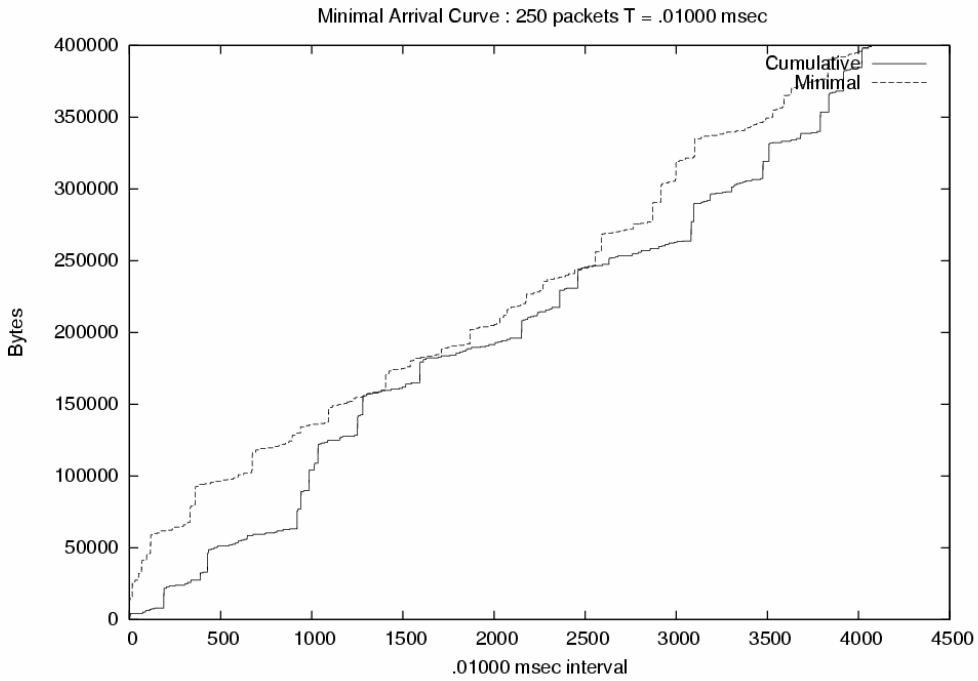
We captured the daily traces from WIDE data repository for 7 days of a week (1^{st} January 2004 to 8^{th} January 2004), and then 5 times in a weekly basis (15^{th} January 2004, 22^{nd} January 2004, 29th January 2004, 5^{th} February 2004 and 12^{th} February 2004).Each time the data was collected for 15 minutes of trace (from 14:00 Hrs to 14:15Hrs) over an USA-JAPAN pacific link. The *tcpdump* files collected were of huge size (~300Mb) so we extracted the various no of initial packets to do our calculations. We choose 4 different initial sizes i.e. 250 packets, 500packets, 50000packets and 20000packets. For 250 packets we choose the time interval of 10 Microseconds and the σ size of 5 Micro seconds and calculated the cumulative arrival and minimum arrival curves as shown below. Similarly we choose different time intervals and different σ sizes for other packets and plotted all the cumulative and minimum arrival curves. For first 500 packets we considered three time intervals (100 & 1000 micro seconds & 10 ms) and three σ window sizes (5 and 500 micro seconds & 5ms) respectively. For first 5000 packets we picked the three time intervals as 1ms,10ms and 100ms and σ window sizes of 0.5ms, 5ms and 50ms respectively. Similarly for first 20,000 packets we considered the time intervals as 50ms and 500ms and the σ sizes of 5ms and 50ms respectively.

We have calculated the autocorrelation functions the similar way. But here we have assumed data for 4 different days (1^{st} February,5^{th} February,12^{th} February and 15^{th} February' 2004) and for 3 different size of initial packets (500,1000 and 10,000). We have calculated the mean and standard deviation and then from those values we finally calculated the auto correlation functions and then plotted it.

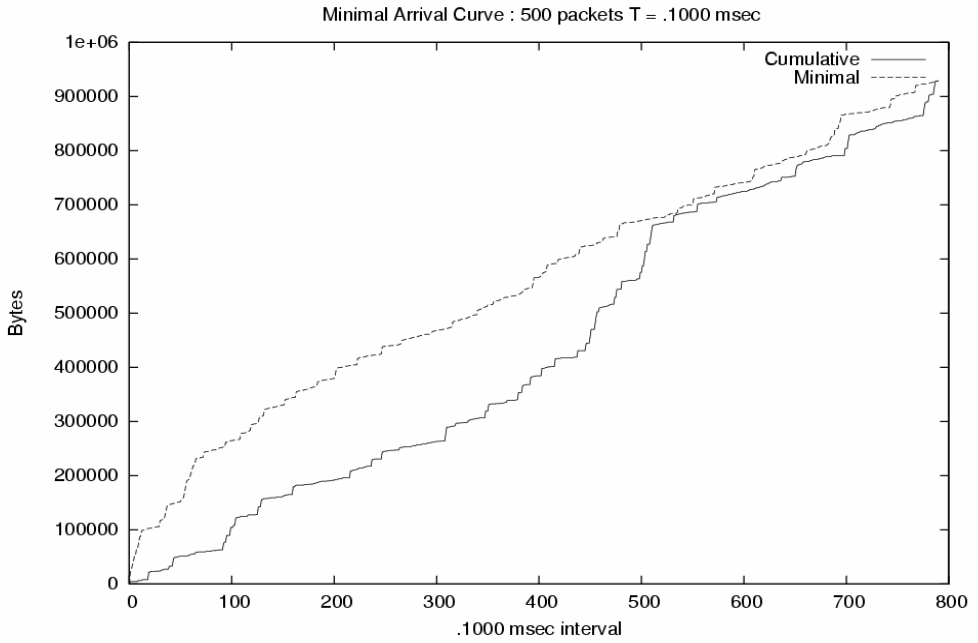
4. Results & Observations:

4.1 Cumulative and Minimum Arrival curves of Day1 (1st January 2004) of various initial packets, various time Intervals and various window sizes:

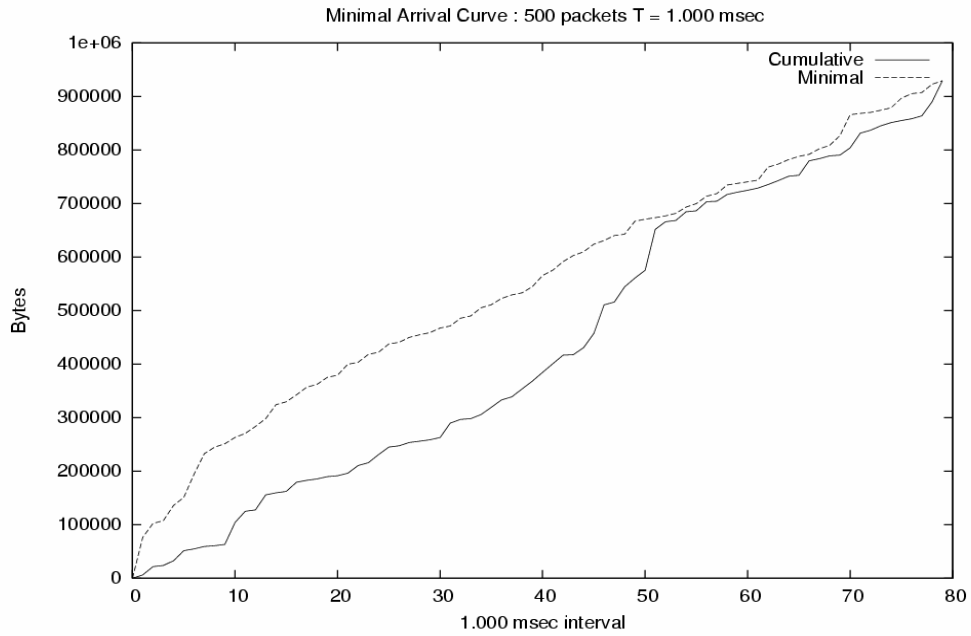
1.dump.sample.minArr_250_0.00001.0.000005.ps



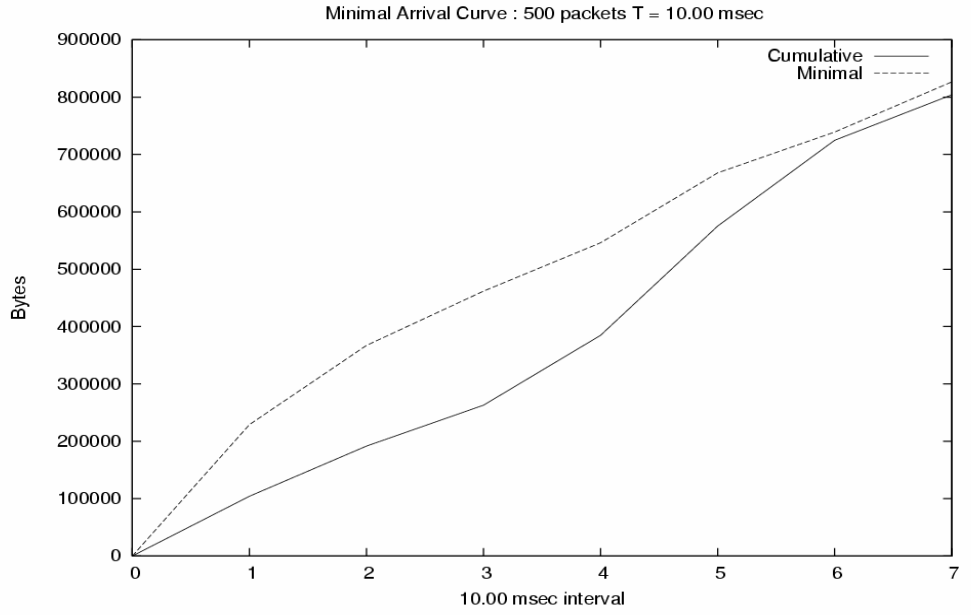
1.dump.sample.minArr_500_0.0001.0.000005.ps



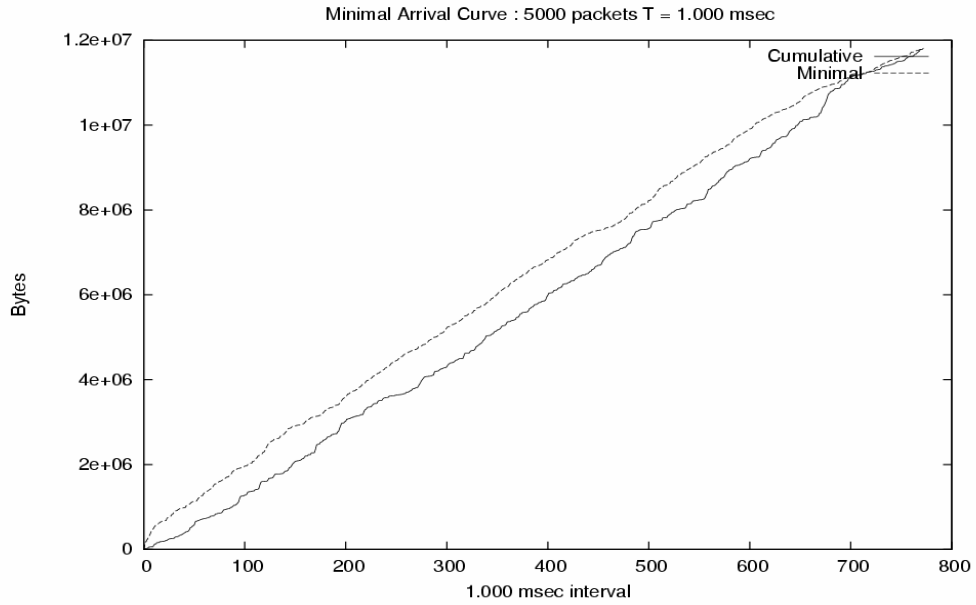
1.dump.sample.minArr_500_0.001.0.0005.ps



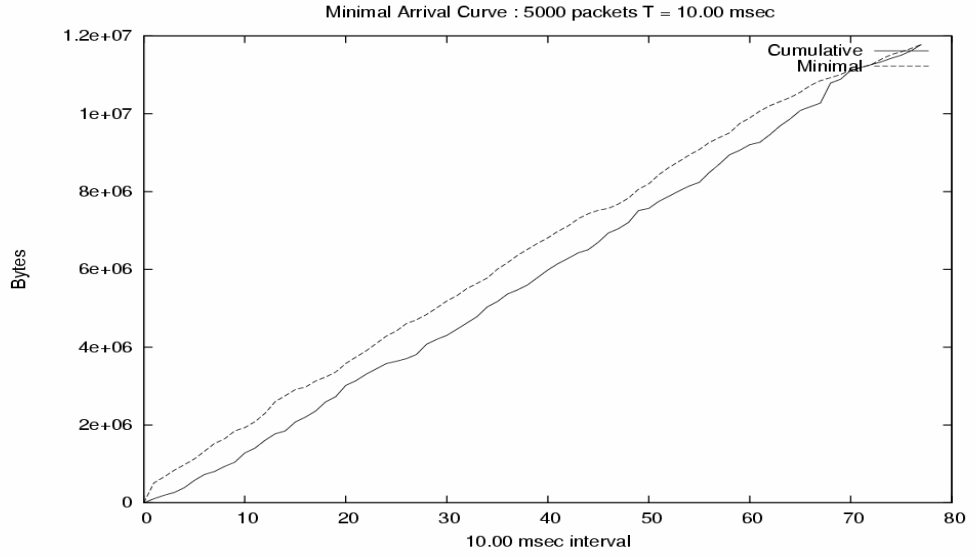
1.dump.sample.minArr_500_0.01.0.005.ps



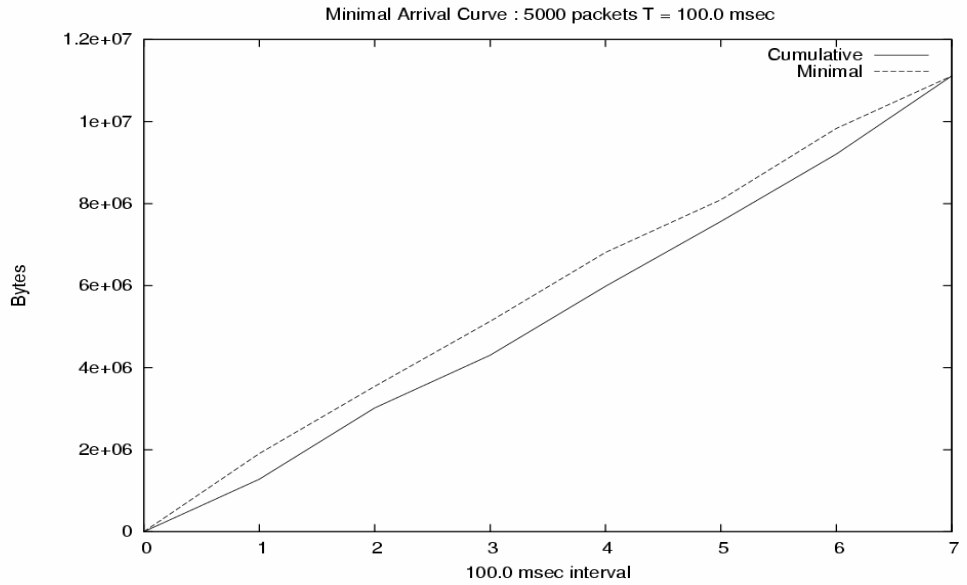
1.dump.sample.minArr_5000_0.001.0.0005.ps



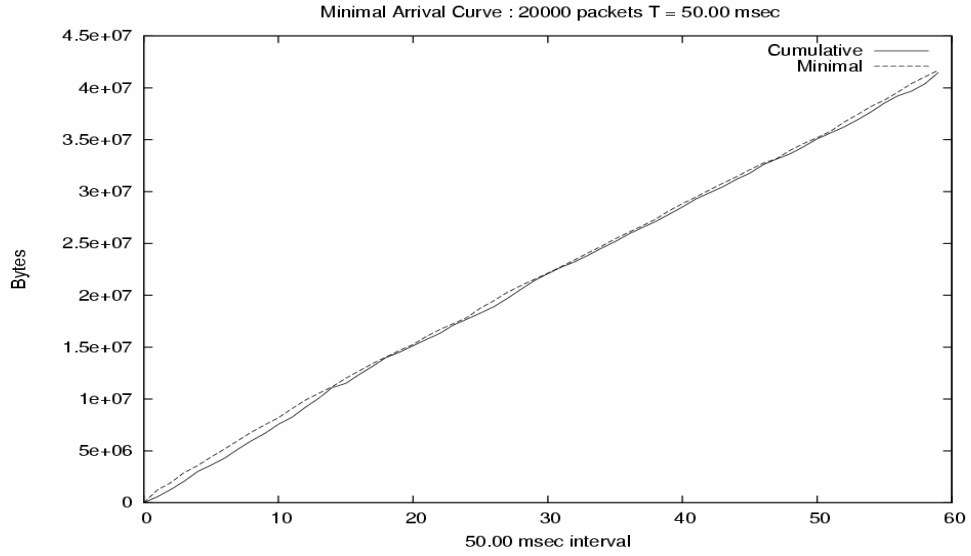
1.dump.sample.minArr_5000_0.01.0.005.ps



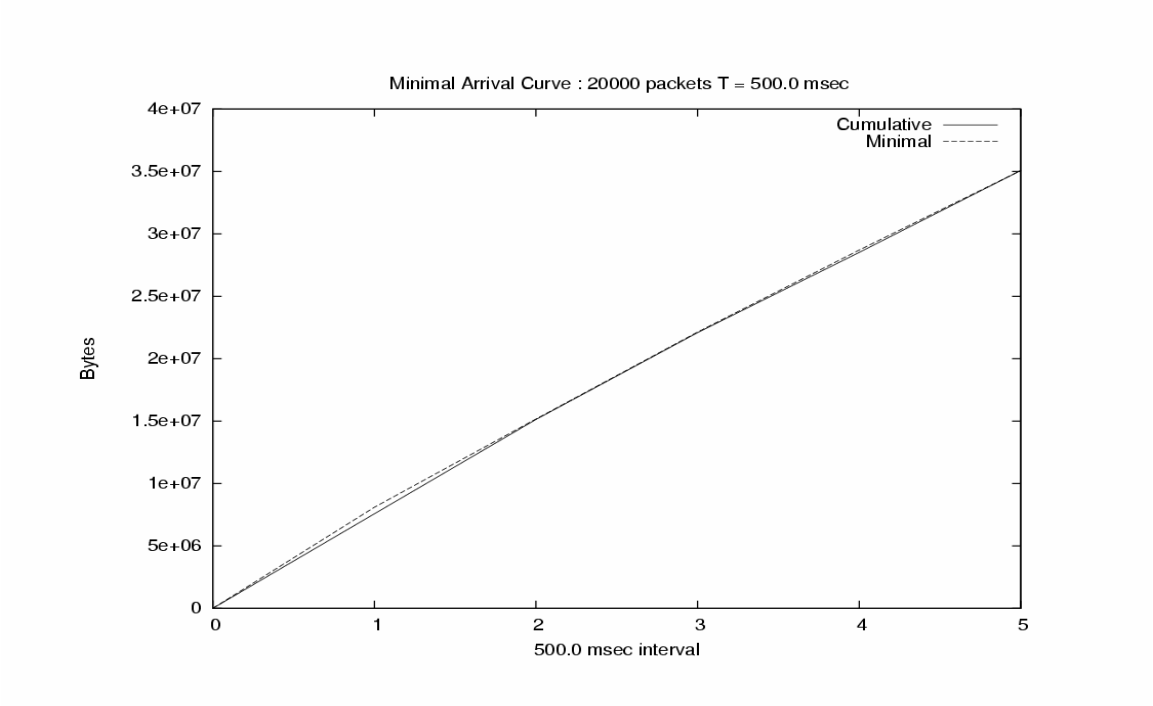
1.dump.sample.minArr_5000_0.1.0.05.ps



1.dump.sample.minArr_20000_0.05.0.005.ps



1.dump.sample.minArr_20000_0.5.0.05.ps



4.1.1 Observations:

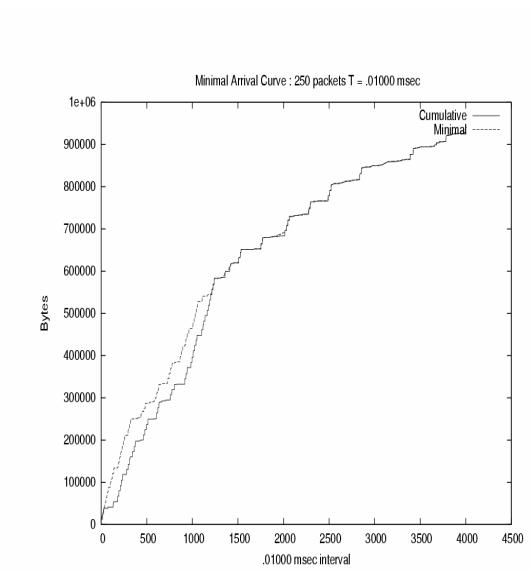
There are 9 graphs shown above. The names of the graphs are self explanatory. For example the last graph's name is 1.dump.sample.minArr_20000_0.5.0.05.ps. This graph is meant for 1st January 2004 daily traces; for first 20,000 packets; with a time interval of 0.5 seconds and with a σ size of 0.05 seconds. Similarly all above graphs can be identified properly.

We can observe here that as we increase the no of initial packets from 250 to 20,000 the smoothness of curves is increasing. For first 250 packets the cumulative arrival curve is a proper stepped one having proper edge value as each time some packets arrive. It carries the minute details of the network characterization of traffic models. Also the minimum arrival curve signifies properly that it constraints the cumulative arrival curve. It can never be lower than cumulative arrival curve.

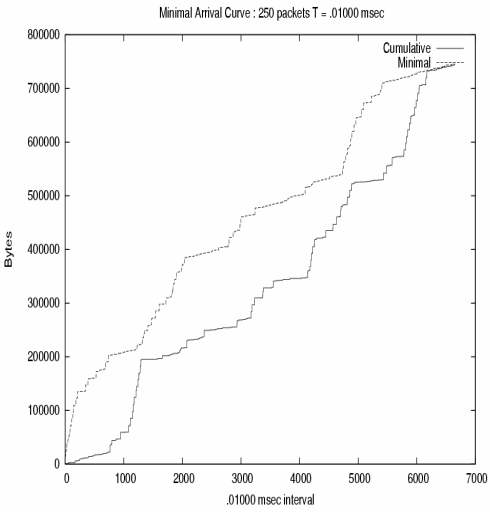
The second observation is that, while we increase our time interval keeping the extracted no of packets as same, we are getting the cumulative arrival curve more straight or smoother. This is also obvious as because if the time interval is less then the role of bits arrival rate will be significant.

The third observation is that when the σ window size decreases, the minimum arrival curve gets closer to the cumulative arrival curve. This is because from the formula we can see that as σ increases we can get the minimum arrival curve closer to the cumulative arrival curve.

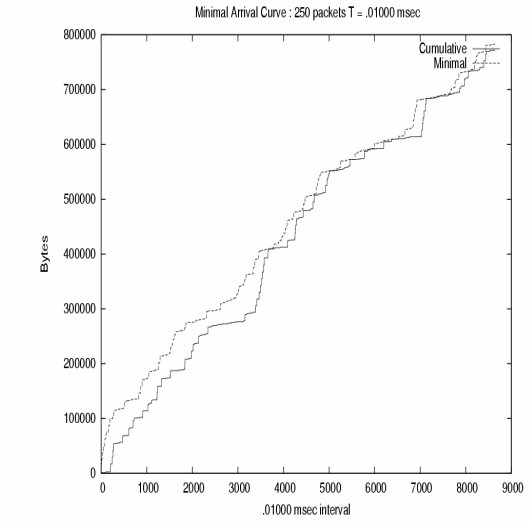
4.2 A comparison between Cumulative and Minimum Arrival curves of different days of initial 250 packets, time Interval of 0.00001 seconds and window size of 0.000005seconds:



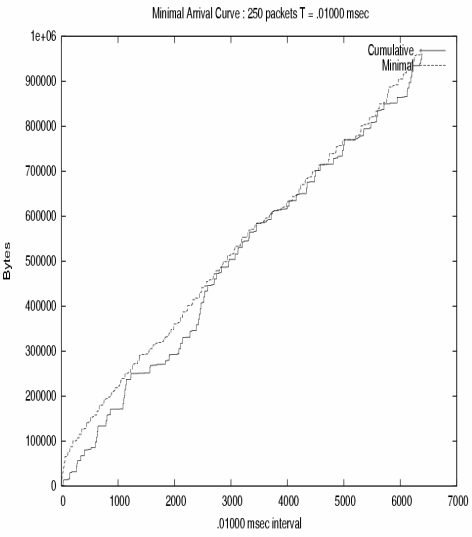
2ndjan.minArr_250_0.00001.0.000005



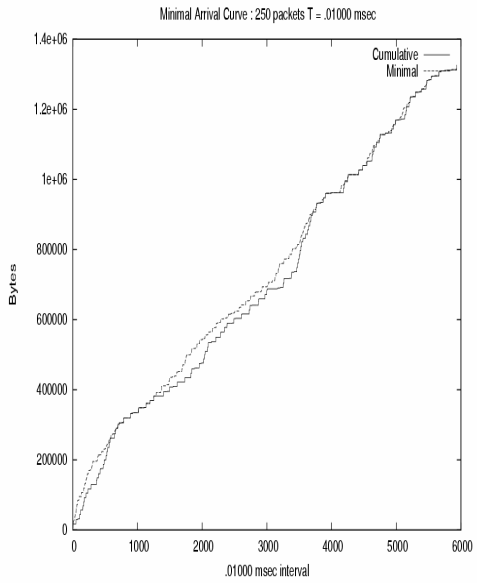
3rdjan.minArr_250_0.00001.0.000005



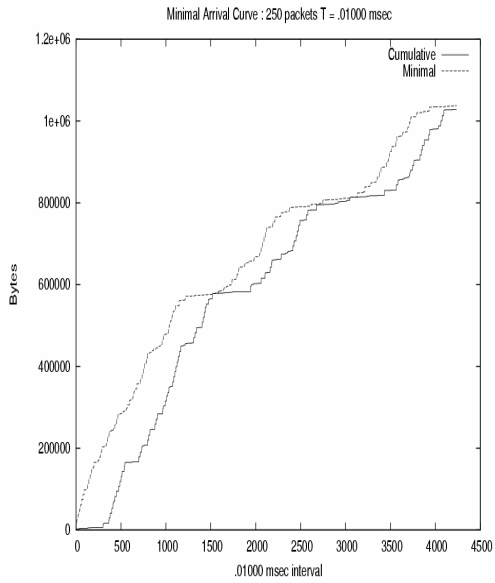
4thjan.minArr_250_0.00001.0.000005



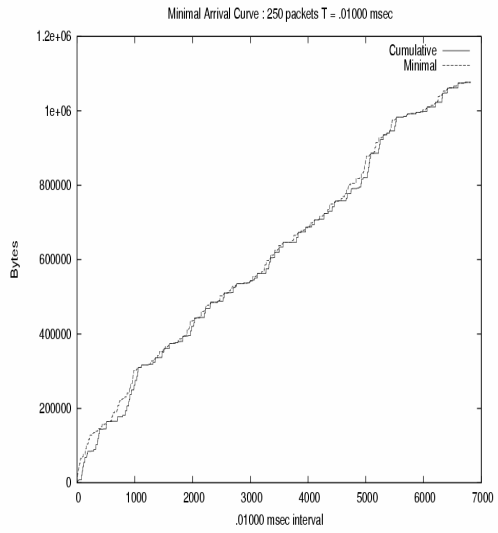
5thjan.minArr_250_0.00001.0.000005



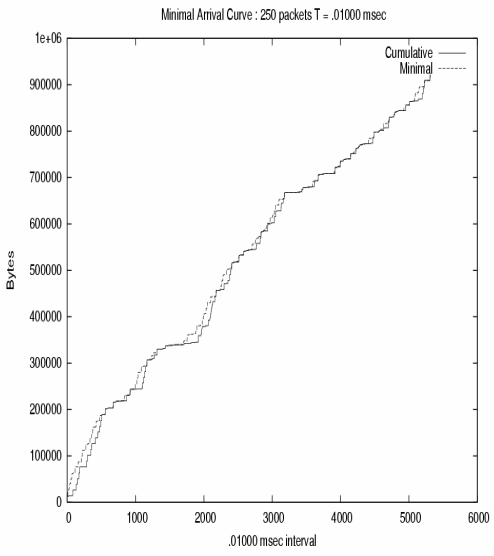
6thjan.minArr_250_0.00001.0.000005



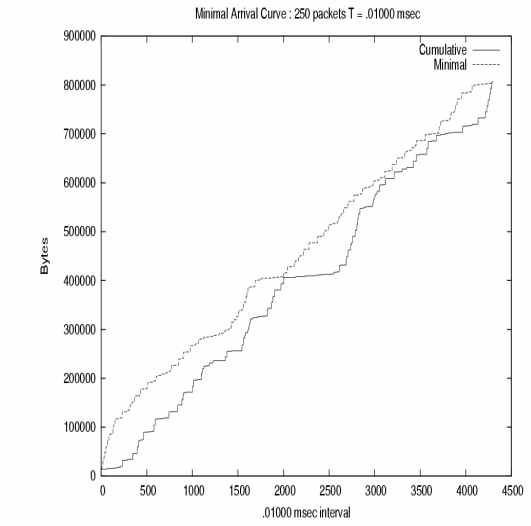
7thjan.minArr_250_0.00001.0.000005



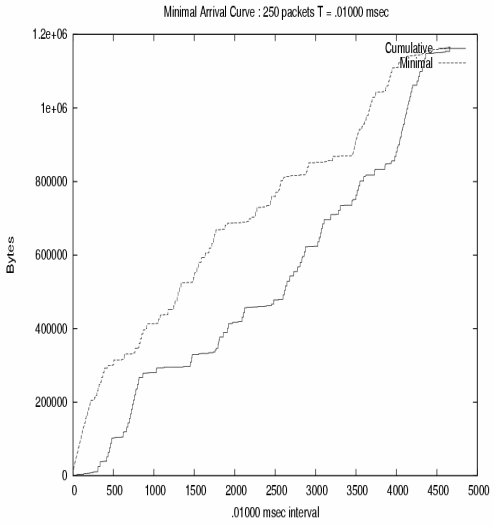
8thjan.minArr_250_0.00001.0.000005



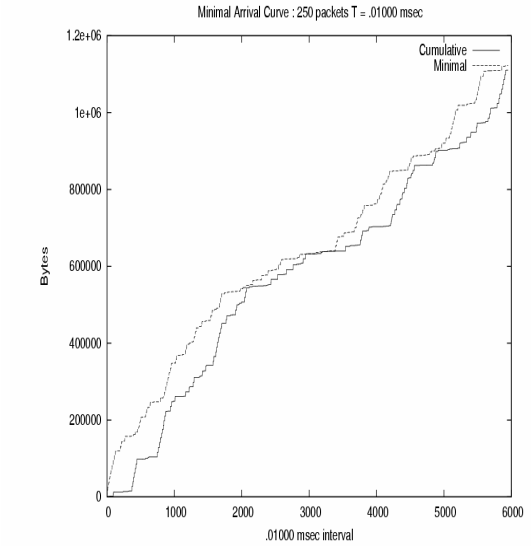
15thjan.minArr_250_0.00001.0.000005



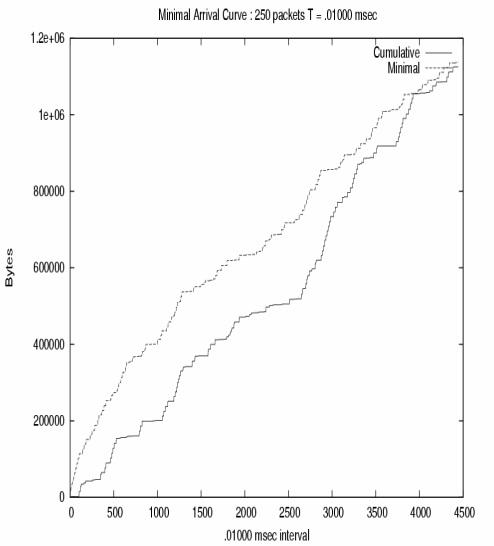
22ndjan.minArr_250_0.00001.0.000005



29thjan.minArr_250_0.00001.0.000005



5thfeb.minArr_250_0.00001.0.000005



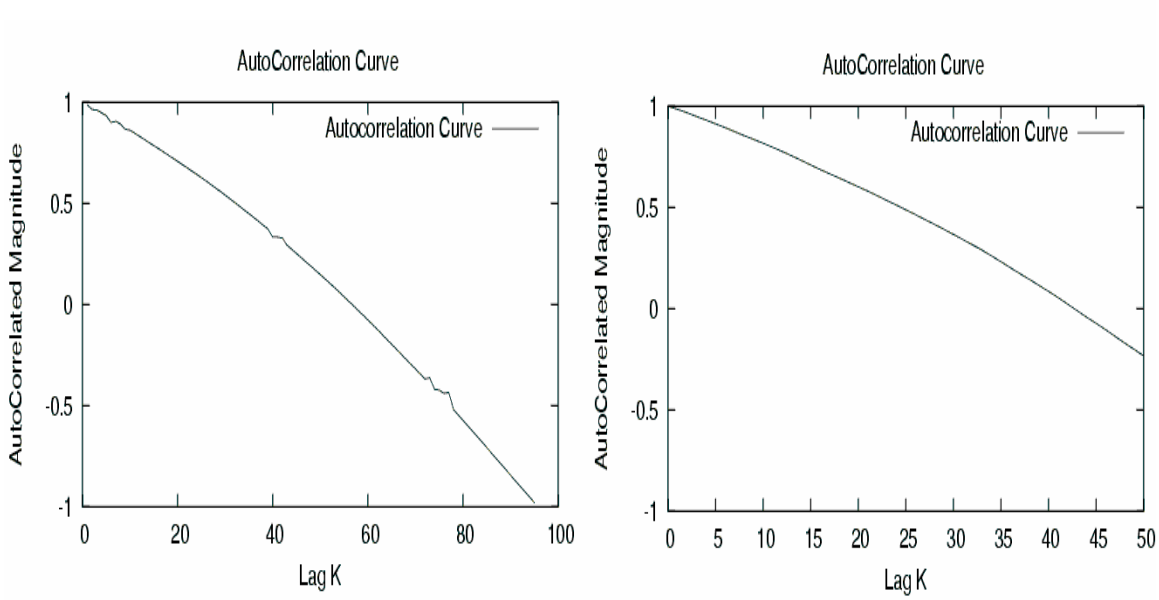
12febminArr_250_0.00001.0.000005

4.2.1 Observations:

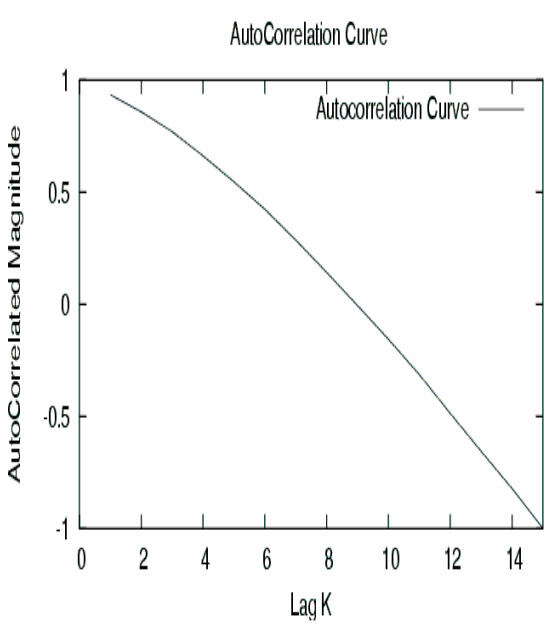
There are 12 graphs shown above. These graphs are of initial 250 packets, of the time interval 10micro seconds and σ window sizes of 5 micro seconds. We have shown the same graphs for different daily traces. As we trace on 13 different days, the 12 traces have been shown here as the first one has already been shown in the previous section. From these graphs we can observe that the cumulative and minimum arrival curves are unpredictable in a time scale. Comparing the graphs of all the days of the 1st week of January 2004 we can say that, the dependency and predictions of data arrival rate on any day is not significantly affected by other days. For example in the graph 2ndjan.minArr_250_0.00001.0.000005, the cumulative arrival curve is increasing suddenly but in the graph 3rdjan.minArr_250_0.00001.0.000005, the cumulative arrival curve is increasing step by step basis.

Comparing the graphs on a weekly basis (8thjan.minArr_250_0.00001.0.000005 and 15thjan.minArr_250_0.00001.0.000005) we can see that the cumulative curve in the second one does have some spikes whereas the first one shows slightly more smoothness. Seeing the last 4 graphs we can predict that there lies some dependcies between the data rates in a network traffic on a timely scale provided the time scale is large. So we may be sure of a long range dependency in a internet networking.

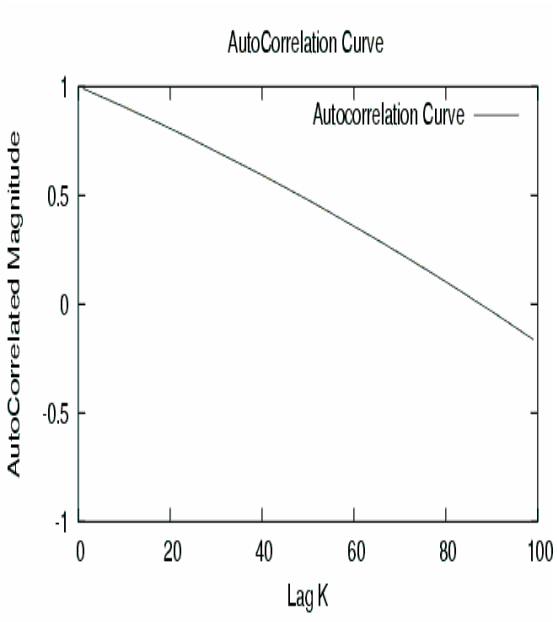
4.3 Auto Correlation Functions of different no. of initial packets over cumulative and Minimum arrival curves:



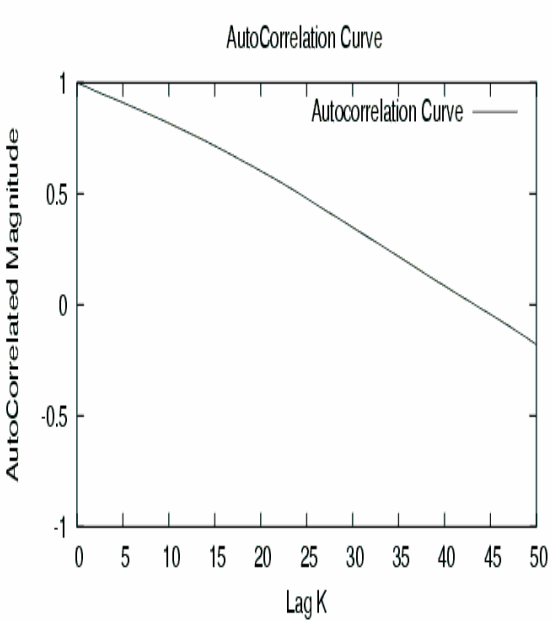
0201cumul10000 0205cumul500



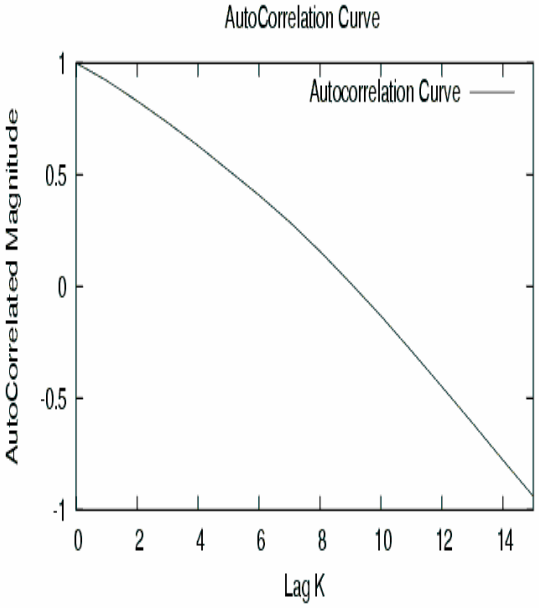
0205cumul1000



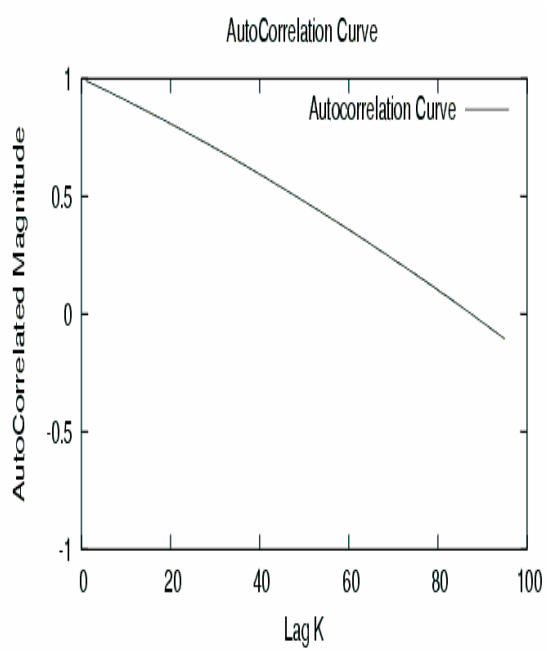
0205cumul10000



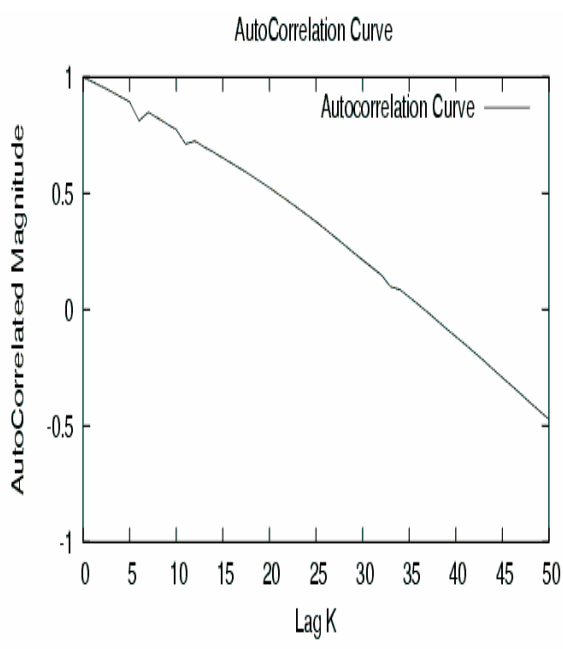
0205minarriv500



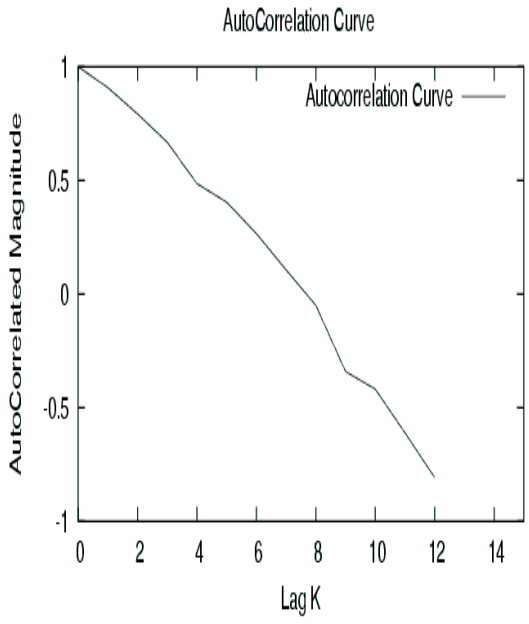
0205minarriv1000



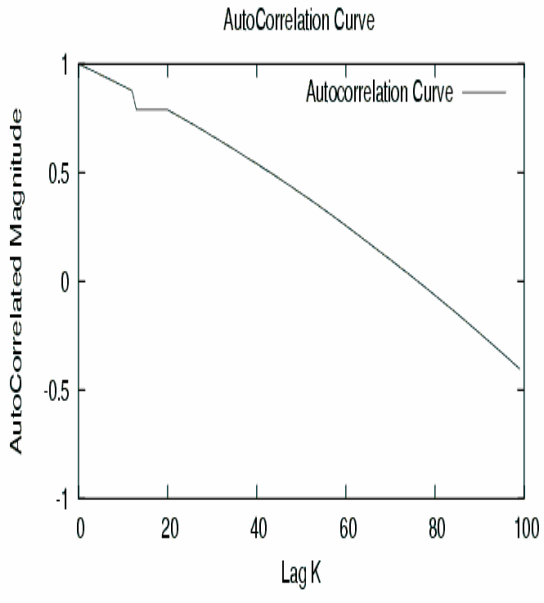
0205minarriv10000



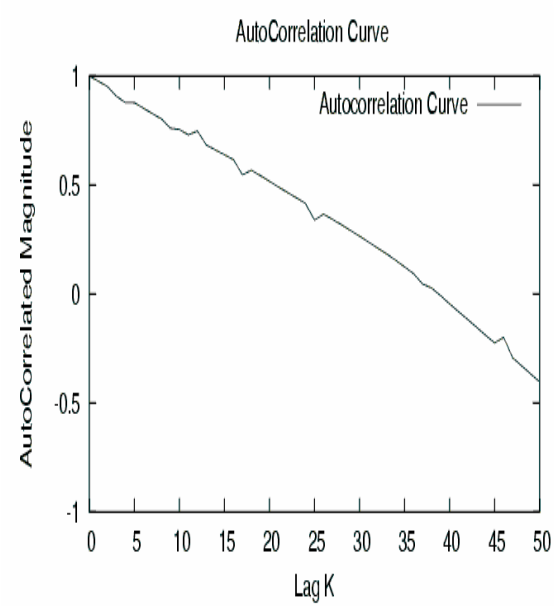
0212cumul500



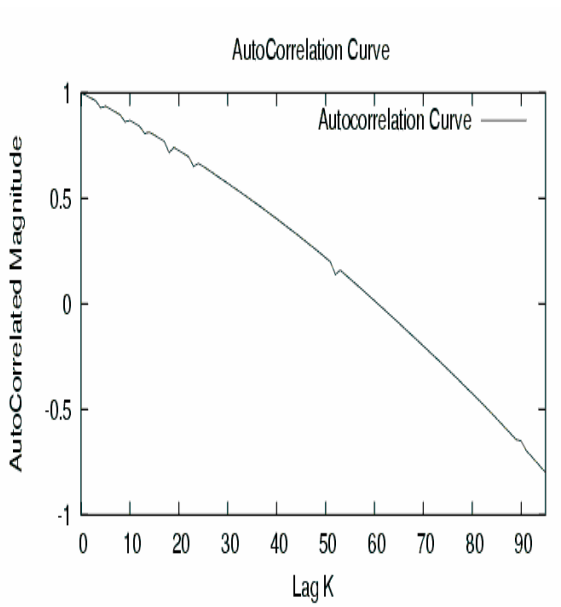
0212cumul1000



0212cumul10000



0215cumul500



0215cumul10000

4.3.1 Observations:

There are 12 graphs shown above. The first four digits of the graphs' names are defined according to the month and date of the year 2004. The next tag symbolizes whether the autocorrelation function is calculated over cumulative arrival curve or minimum arrival curve. The last tag says the no. of the initial packets we considered.

In all the above figures the x-axis shows the no of lags for the defined no. of packets we considered and the y-axis shows the auto correlated magnitude. For example in the graph "0215cumul10000";the last one, the autocorrelation function was calculated for 95 lags for 10,000 consecutive packet arrivals for the packet sizes and inter arrival series. The sizes series consists of the actual packet sizes as individual packets arrive; the interarrival series consists of the timestamp differences between consecutive packets. Apart from some limited correlation at small time lags, sizes and interarrivals are not correlated.

We can mark some spikes in the curves. That shows that there exists some periodicity over the long range dependency.

5. Conclusions:

In this paper, we calculated some of the characterization parameters of internet networking by examining a number of traces of internet traffic. We studied the mean off sets and smoothness of the data arrival rate over a period of time and also we studied the correlation of these data arrival rates on a time basis. We reached at a certain point that the ongoing pattern of internet evolution may potentially affect the future characteristics of its traffic. And also we learned that there is some short of periodicity over the long range data. Based on the traces from WIDE backbones, we found that up to micro second scales traffic is well characterized by our network calculus theory.

6. References:

1. Network Calculus; A theory of Deterministic queuing systems for the internet by Jean yves le Boudec & Patrick Thiran.

2. Communication Networking; by Kumar & Manjunathan; Morgan Kaufman.

3. A Nonstationary Poisson View of Internet Traffic: Thomas Karagiannis,Mart Molle, Michalis Faloutsos.

4. Long Range Dependence; Ten Years of Internet traffic Modeling; by Thomas karagiannis, Mart Molle and Michalis Faloutsos

5. WIDE project web page ; http://www.wide.ad.jp/

7. Appendices:

7.1 The source code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
typedef struct n{
        double timestamp;
        double start,end;
        float propagationTime;
        int size,type;
struct        n * back;
struct        n * next;
}node;
double maxTime,minTime;
node *handle;

float R(double,double);
void plot_cumulative_arrival_curve(double);
void arrangeBins(double);

main(int argc,char *argv[])
{

        double i,j,timestamp1,timestamp2;
        double bin,maxTbin,maxVbin,binT,binV,loop1,loop2;
        unsigned int ui;
        int k,l,pktSize;
        node *queue,*ptr;
        int retcode;
        FILE *input,*fp;
        input = fopen(argv[1],"r");
        binT = atof(argv[2]);
        binV = atof(argv[3]);

        if(!input)
                exit(-1);
        queue=NULL;
        fp = fopen("minarr.dat","w+");
        do
        {
        fscanf(input,"%lf\t%d\n",&timestamp1,&pktSize);
        ptr =  ( node *) malloc(sizeof(node));
        ptr->timestamp = timestamp1;
        ptr->size = pktSize*8;
        ptr->propagationTime = (float)ptr->size/10000000;

        if(queue==NULL)
        {
                queue = handle = ptr;
                ptr->back = NULL;
                ptr->next = NULL;
        }

        ptr->back = queue ;
        queue->next = ptr;
```

```
        queue = ptr;
        ptr->next = NULL;

        }while(!feof(input));

        ptr = handle;

        while(ptr)
        {
//        printf("time = %f packet size = %d transmit time = %f
uSec\n",ptr->timestamp,ptr->size,(float)ptr->propagationTime);
        maxTime = ptr->timestamp;
        ptr = ptr->next;
        }
        fclose(input);
        minTime = handle->timestamp;

        // R O R = sup v>=0  R(T+v) - R(v)
        //
        float var,tmpmax=0.0;

        maxTbin = (maxTime - minTime) / binT;
        maxVbin = (maxTime - minTime)/binV;
        ptr = handle;

        plot_cumulative_arrival_curve(binT);
//        arrangeBins( binT);

        loop1=0.0;
        loop2=0.0;
        printf("before entering the for loop\n");
        printf("%f\t%f\t%f,\t%f\t%f\n",binT,binV,maxTime-
minTime,maxTbin,maxVbin);
                fprintf(fp, "%f\t%f\n",0,0);
        while(loop1 < maxTbin)
        {

                while(loop2 < maxVbin)
                {
//                        printf("for T = %lf\t V
=%lf\n",loop1*binT,loop2*binV);
                        var = R(loop1*binT,loop2*binV);
                        if(var > tmpmax)
                        {
                                tmpmax = var;
                //        printf("tmppax = %f\n",tmpmax);
                        }
                        loop2 +=1.0;
                }

                printf( "%f\t%f\n",binT*loop1,tmpmax);
                fprintf(fp, "%f\t%f\n",loop1,tmpmax);

                tmpmax = 0;
                loop1+=1.0;
                loop2=0;

        }
```

```
}

float R(double T , double V)
{
node *ptr;
ptr = handle;
float total_arrival=0.0,t_plus_v=0.0,v=0.0;
if(V<T)
{
        while(ptr && ptr->timestamp < (minTime + V) )
        {
                v+=ptr->size;
                ptr=ptr->next;

        }
        t_plus_v = v;
        while(ptr && ptr->timestamp < (minTime + V + T) )
        {
                t_plus_v+=ptr->size;
                ptr=ptr->next;

        }

        return(t_plus_v - v);

}
else
{
        while(ptr && ptr->timestamp < (minTime +V) )
        {
                v+=ptr->size;
                ptr=ptr->next;

        }
        t_plus_v = v;
        while(ptr && ptr->timestamp < (minTime + T + V) )
        {
                t_plus_v+=ptr->size;
                ptr=ptr->next;

        }

        return(t_plus_v - v);
}
}

void plot_cumulative_arrival_curve(double binSize)
{
        double i=0.0,total = (maxTime - minTime) / binSize;
        float arrival=0.0;
        node *ptr;
        ptr= handle;
        FILE *fp;
        fp = fopen("cumulative.dat","w+");
        for(i=0.0;i<total;i+=1.0)
        {
                while(ptr && ptr->timestamp < minTime + i*binSize)
                {
```

```
                        arrival +=ptr->size;
                        ptr=ptr->next;
                }
                fprintf(fp,"%f\t%f\n",i,arrival);

        }

        fclose(fp);
}


void arrangeBins(double binSize)
{
node * ptr;
ptr = handle;
float data=0.0;
FILE *fp;
fp = fopen("hurst_input.dat","w+");

double i,totalBins = (maxTime - minTime) / binSize;
        for(i=0.0;i<totalBins;i = i+1.0)
        {
                while((ptr->timestamp +ptr->propagationTime ) <
(minTime + i*binSize))
                {
                        if(ptr->timestamp < (minTime + (i-1)*binSize))
                        {
                                //add remaining portion of the packet
to this bucket

                data+=((ptr->timestamp + ptr->propagationTime) -
(minTime + (i-1)*binSize))*10000000;
                        }
                        else{
                                data += ptr->size;
                                //add whole packet size to this bin
                        }
                ptr = ptr->next;
                }
                //now out of while loop
                //check if the current packet has some portion in this
bin
                if(ptr && (ptr->timestamp < minTime + i*binSize))
                {

                data+=((minTime + i*binSize)-ptr->timestamp)*10000000;

                }

                //okay now thi bin contains exact amount of data
received in thst time limit
                //output this data against this bin number
                fprintf(fp,"%f\n",data);
                data = 0.0;
        }

fclose(fp);

}
```

7.3 The script for plotting the graphs:

```
#!/bin/sh
outfile=outfile
T=$2
V=$3
num=$T
thousand=1000
for i in `ls *.sample`
do
packets=$1
rm -f $outfile
echo "doing for file $i"
tcpdump -e -q -n -N -O -c $packets -r $i > tmp
awk '{print $1, $7}' tmp > tmp1
sed -f rem.sed tmp1 > tmp2
awk '{print $3, $4}' tmp2 > $i.trace
rm -f tmp tmp1 tmp2
num=`echo $T \* $thousand|bc`
echo "bin = $T delta = $V "
./a.out $i.trace $T $V
echo "set output \"minArrival.ps\" " >> $outfile
echo "set terminal postscript" >> $outfile
echo "set title \" Minimal Arrival Curve : $1 packets T = $num msec\" "
>> $outfile
echo "set xlabel\"$num msec interval\"" >> $outfile
echo "set ylabel\"Bytes\"" >> $outfile
echo "plot \"cumulative.dat\" using 1:2 title 'Cumulative' with lines
" >> $outfile
echo "set output \" minArrival.ps\" " >> $outfile
echo "set terminal postscript" >> $outfile
echo "replot \"minarr.dat\" using 1:2 title 'Minimal' with lines  " >>
$outfile
gnuplot outfile
rm -f outfile
mv -f cumulative.dat readings/$i.cum_$1_$T.$V.dat
#mv -f hurst_input.dat readings/$i.hurst_$1_$T_$V.dat
mv -f minarr.dat readings/$i.minarr_$1_$T.$V.dat
mv -f minArrival.ps graphs/$i.minArr_$1_$T.$V.ps
mv -f $i.trace trace/$i.$1Pkts.trace
rm -f tmp tmp1 tmp2
done
```

7.4 The Root script:

```
./clean
./doAll.sh 500  0.0001 0.000005
./doAll.sh 500  0.001  0.0005
./doAll.sh 500  0.01   0.005
./doAll.sh 5000 0.001  0.0005
./doAll.sh 5000 0.01   0.005
./doAll.sh 5000 0.1    0.05
./doAll.sh 20000 0.05  0.005
./doAll.sh 20000 0.5  0.05
```

7.5 Autocorr Code:

```
/* Program to calculate autocorellation function for traces  */
/* formula used, C(k)=(1/n){Sum(Y(i)-Y(mean)) Y(i+k)-Y(mean)}*/
/*                      ------------------------------------*/
/*                           (1/n) Sum {pow(Y(i)-Y(mean),2}    */
/* Numerator varies from k to N-k-1, Denominator, 0 to N-1   */

#include <stdio.h>  //basic file definitions
#include <stdlib.h> //compilation with G++, build 3.06
#include <math.h>
char *ProgNM;
#define MAX 5000
main(int argc, char **argv)
{
        int Var1,var2;
        int c,numpts;
        FILE *fp, *fopen(); //define the file pointers
        extern int Iopt;
        extern char *argOpt;
        double sum[2],avg[2],sumsq[2],std[2],max[2],min[2];

double dat[MAX][2];
        double sqrt();
        double temp,cor;
        int tau;
        float deltat=-1;

        tau = 100;
    while ((c = getopt(argc, argv, "ht:d:")) != EOF)
        switch (c)
        {
        case 'h':

//specify the command line options

fprintf(stderr,"input should be one column of data\n\n");
fprintf(stderr,"Usage: cor [<] datafile\n");
fprintf(stderr,"\nOptions:\n");
fprintf(stderr," -h   : this help option\n");
fprintf(stderr," -d # : set maximum delay for autocorrelation
[%d]\n",tau);
fprintf(stderr," -t # : set delta-t between each point [1.0]\n");
               exit(1);
               break;
        case 'd': tau = atoi(argOpt);
                        break;
        case 't': deltat = atof(argOpt);
                        break;
        }

    argc -= (Iopt-1) ; argv += (Iopt-1) ;
    fp = (argc > 1) ? fopen(*++argv, "r") : stdin;

        numpts = 0;
        for (Var1=0;r<2;r++)  {
                sum[Var1] = 0.0;
                sumsq[Var1] = 0.0;
                max[Var1] = -9.999e99;
                min[Var1] = +9.999e99;
        }
```

```
while (fscanf(fp,"%lf",&dat[numpts][0]) == 1)  {
                for (Var1=0;Var1<1;Var1++)  {
                        sum[Var1] = sum[Var1] + dat[numpts][Var1];
                        sumsq[Var1] = sumsq[Var1] +
dat[numpts][Var1]*dat[numpts][Var1];
                        max[Var1] = (max[Var1] > dat[numpts][Var1]) ?
max[Var1] : dat[numpts][Var1];
                        min[Var1] = (min[Var1] < dat[numpts][Var1]) ?
min[Var1] : dat[numpts][Var1];
                }
                ++numpts;
                if (numpts >= MAX)  {
                        fprintf(stderr,"too many data points, limit is
%d\n",MAX);
                        exit(1);
                }
        }

        fprintf(stderr,"   avg        std        max         min
number\n");
        for (r=0;r<1;r++)  {
                avg[Var1] = sum[Var1]/numpts;
                std[Var1] = sqrt((sumsq[Var1]/numpts)-
avg[Var1]*avg[Var1]);
                fprintf(stderr,"%10g %10g
%10g",avg[Var1],std[Var1],max[Var1]);
                fprintf(stderr," %10g %10d\n",min[Var1],numpts);
                std[Var1] = 1.0/std[Var1];
        }
        cor = 0.0;
        for (var2=0;var2<numpts;var2++)
                dat[var2][0] = (dat[var2][0]-avg[0])*std[0];

        for (Var1=0;Var1<tau;Var1++)  {
                cor = 0.0;
                for (var2=0;var2<(numpts-Var1);var2++)
                        cor += dat[var2][0]*dat[var2+Var1][0];
                cor = cor / ((double)(numpts-Var1));
                if (deltat<0.0)
                        printf("%d %g\n",Var1,cor);
                else
                        printf("%f %g\n",((float)Var1)*deltat,cor);
        }
        exit(0);
}
```