1 Introduction

It is known that a wide variety of physical, biological and social systems show power-law correlations in space and time (the 1/f noise) [1, 2, 3, 4, 5, 6, 7]. However the characterization and understanding of these phenomena have remained difficult problems. No established explanation of a $1/f$ PSD (Power Spectrum Density) is not known so far. One tendency is that we can observe $1/f$ fluctuations when the system has intermittency[1, 8, 9].

As a case in social systems, we notice the human activity in Internet[10], international computer network, which is growing day by day. In this network a lot of people everyday use Mail[10, 11], Ftp[10], Distributed Information Services[12] e.g. Wais, Gopher, WWW... and participate in Netnews, Mailing Lists[13] and so on. In this letter we study the traffic of a **Mailing List** where the traffic implies the number of mails delivered through it. Our experience tells us that the traffic has intermittency such as "I receive 10 mails almost everyday, 30 mails today but none yesterday". We observe the time sequence of a traffic to show the $f^{-\mu}(\mu \sim 1)$ PSD.

$1/f$ fluctuations are also considered as a kind of phenomena that the system keeps the memory over long time to show non-white noise in the low frequency range. We propose a model which has a reservoir to memorize system configurations. A set of information can describe our system but we have spare information in the reservoir. The development procedure of our simulation is similar to genetic algorithm[14]. We develop our system using both mutations and crossovers for data in the reservoir. Our model shows a power tail, almost $1/f$, at sufficiently low frequencies in the PSD when mutations and crossovers happen rarely.

Our data is based on the record of a Mailing List over two years. It shows the intermittent time sequence, which seems to lead to $1/f$ fluctuations. We also propose a model similar to genetic algorithm to show the $1/f$ fluctuations. This result suggests that incomplete shuffling of the initial system information generates $1/f$ fluctuations.

2 Internet, Mailing Lists and Sending Mail

"The Internet is a collection of thousands of networks linked by a common set of technical protocols which make it possible for users of any one of the

networks to communicate with or use the services located on any of the other networks."[15]

The Mail system is one of the most fundamental communication tools [11] in Internet. A mailing list is an email address that stands for a group of people rather than for an individual. Mailing lists are usually created to discuss specific topics. Anybody interested in that topic, may (usually) join that list.

As a model of a mailing list, consider a set of variables $c(i)$ where i runs from 0 to N. $c(i)$ is an abstract object to stand for a **client**. All $c(i)$ are connected each other. Each $c(i)$ is an auto-monic object with links to $c(j)(i \neq j)$ and replies mails. Define $c(0)$ as a server of the mailing list. Technically speaking, each mail from $c(i)(i = 1..N)$ is delivered via $c(0)$ to all $c(j)(j \neq i)$. It is virtually the same as that all $c(i)(i = 1..N)$ are connected and a mail from $c(i)$ is directly sent to $c(j)(i \neq j)$. This is all of the mechanism of a Mailing List. $c(0)$ is called **mailing list server[13]** or **mailing list manager** which is a utility to deliver received mails to $c(i)$ and usually has additional functions to manage the list e.g. subscription. We can drop $c(0)$ considering the model of mailing lists. Hereafter we use the term ML as **mailing list process**, **server** and **client** as ones mentioned above. We call this model **server client model**[16].

The delivery among $c(i)$ is a mail which has several attributes such that it consists of a header and a body where a header consists of e.g. the address to send, the subject and a body is the main text of the mail. Consider the symbolic representations for them e.g. **mail**, **header**. They are considered as a list where a **list** represents a sequence of zero or more elements (which may be any Lisp objects)[17]. It is schematically represented such as

```
mail
=> (header body)
header
=> (from to subject)
```

where $\Rightarrow$ implies to show the content of the symbol. We can consider the mail has additional attributes e.g. length

```
mail
=> (length header body).
```

Here we drop all attributes of the mail and we notice only the the number of mails delivered among $c(i)$.

3 Observation

Technically speaking, we have only to observe the traffic through the server to investigate ML. The time sequence of the traffic is such as 10, 3, 4, 0, 1, 2, 30, 10, 0, 3, 3, 0, 2, 0, ... The traffic is almost constant but sometimes bursts like a 30, 10, 0 ...

In our case we generate a time sequence from a Mailing List log by setting the unit to three hours. The averaged traffic a day is about 30 mails, that is about 10 thousand a year, and the maximum of a burst is about 100 mails a day. Almost all the traffic are 0 and the period when the traffic per the unit time is less 3 mails covers about 90 % of our observation data totally. The traffic sometimes bursts to $20 \sim 30$ mails abruptly. Such a behavior is known as intermittency[1, 8], which matches our intuition and feeling to our network environment.

We calculate PSD by taking a Fast Fourier Transform [18]. See Fig. 1. The PSD has a typical form to show the white noise in the high frequency and almost $1/f$ dependence as $f \to 0$. We can see some peaks in the white noise region of the PSD where the highest peak corresponds to one day and the lowest corresponds to one week. These peaks reflect human biorhythm. The range where the power tail is seen is the time scale over one week. It is plausible that the long time correlation appears in the scale over one week, since the activity in the scale less than a week must work almost periodically and randomly.

4 Overview on Genetic Algorithm

Forgetting the old time information would correspond to the absence of long time correlation which leads to the white noise in the lowest frequency in PSD. One way to consider $1/f$ fluctuations would be to investigate how a system could store the information over long time since non-white noise in the $f \to 0$ limit implies some correlations over a long period. The mechanism to record and forget the initial system configurations may cause $1/f$

fluctuations. To produce the mechanism we propose a procedure similar to genetic algorithm(GA).

GA is a model of machine learning which derives its behavior from a metaphor of the processes of evolution in nature. This is done by the creation within a machine of a population of individuals represented by e.g. a bit sequence analogous to our gene. The individuals in the population then go through a process of evolution.

GA and the modified ones are well used as an optimization problems. GA is expected to do a global search. The optimization implies genes which fit the environment best survive. As an analogy to the exchange of genetic information at the molecular level, we operate a re-combination on DNA, which is referred as CROSSOVER in GA. CROSSOVER is a re-combination process of individuals which is fitted to the environment. SELECTION is to select which individuals are fitted to the environment. MUTATION plays an another role for the evolution process as stochastic perturbation to genes.

The three operations characterizes GA. Using only mutation is almost the same as a random search. The crossover is an important process in GA to do global searches.

The algorithm of GA is quasi-coded as follows

```
PSEUDO CODE
     start with an initial time
     initialize a usually random population of genes
     evaluate fitness of all initial genes

     while ( until the given condition is satisfied )
     do {
           increase the time counter
           select a sub-population for offspring production
           recombine the "genes" of selected parents
           perturb genes stochastically
           evaluate it's new fitness
           select the survivors from actual fitness
     }
```

5 A model to reproduce mailing lists properties

For further investigation, we propose an abstract model of ML based upon the experience and observations summarized as follows. Consider the evolution of threads where we call a successive series of a subject a **thread**. A thread has several attributes, but we drop superfluous factors and summarize them to two components **it is active or not now** and **how active it is**. It is symbolically written as a list (active-p how-active) where **p** stands for **predicate**. It is convenient to number the thread. We add an attribute **number** to i -th thread.

```
thread
=>(number active-p how-active)
=>(i      active-p how-active)
```

where **active-p** is one bit such that 1 is active, 0 is not and **how-active** is a real number.

Our experience tells us that several threads run simultaneously. Hence it is convenient to consider a list of threads in a bundle. We call the bundle superthread. It is written as

```
superthread
=>( thread-1  thread-2  thread-3     ... )
=>( (1 0 1.0) (2 1 2.3) (3 0 0.2)  ... ).
```

Only one superthread describes the present status of ML in our model. We call it **ML superthread** for convenience. When we consider a set of $c(i)$ mentioned above, ML superthread is bundled with $c(0)$ and it determines reaction of $c(i)(i = 1..N)$.

We could define the distance **D(thread, another thread′)** between thread and thread′ though it is impossible to describe the practical calculation. Each thread in ML superthread has a small distance between each other. We prepare other superthreads which consist of threads with large distances between ones of ML superthread. Hereafter we call the prepared sets **information reservoir** for convenience. Using information reservoir we describe a situation that unusual kinds of topics sometimes become popular.

We can summarize properties of threads as follows.

1. Interest of people is unsettled. Popularity of threads changes e.g. the active 1st thread becomes not-active at some time

   ```
   (1 0 1.0) -> (1 1 1.0)
   ```

2. It could happen a spin-off of threads such that popularity of similar threads **D(new thread, present thread)** ~ 0 changes simultaneously e.g. 1st thread ends and 2nd thread becomes active.

   ```
   (1 1 1.0) -> (1 0 1.0)
   (2 0 1.4) -> (2 1 1.4)
   ```

   A branch that plural threads arise from a thread could happen.

3. It could happen that a new thread far from the present thread ,**D(new thread, present thread)** is large, arises abruptly. We re-combinate a part of threads in ML superthread with threads picked up among **information reservoir** to create a new ML superthread. We use a GA-wise procedure mentioned below as the mechanism of this re-combination[14].

Consider a set of $c(i)$(server client model) again. Each $c(i)$ sends back reply mails according to a function

```
if active thread (active-p is 1)
        f(random-number, how-active)
else
        0
fi
```

for each thread in the superthread. For simplicity the function f() is just a product here. Only active threads contribute to the traffic of ML.

Our simulation performs the following procedure in one step and repeats this to the total 2^{13} or 2^{14} steps. Firstly we initialize all superthreads for ML and information reservoir randomly. For simplicity we do not change values of how-active variables once the values are set initially but each active-p and combinations of threads vary through the whole simulation. It implies the extent of popularity of a thread is almost the same irrespective of the ages. We

determine the total traffic based on ML superthread and clients mentioned above. In the last of one step we introduce two procedures, mutations and crossovers between ML and information reservoir regarding a superthread as a gene.

Mutation is to make a new thread (active-p how-active) from present one stochastically for each thread. Mutation changes the popularity of threads stochastically, so this is a random process in essence. Mixing information by only mutation is the same as a random process to show only f^0 PSD, which implies that the system forgets the memory in old time.

We apply crossover to the superthreads for mixing the system. The crossover binds the present system to the initial system via information reservoir. The crossover operation is to re-produce a new superthread combining parts of each of two **parent** superthreads. One of two is the superthread of ML and the other is a superthread selected randomly among ones in information reservoir. We can describe crossover as the following pseudo code.

```
PSEUDO CODE for CROSSOVER

Superthread of ML                Superthread in information reservoir
(                                (
(1 active-p how-active)           (#1 active-p how-active)
(2 active-p how-active)           (#2 active-p how-active)
(3 active-p how-active)           (#3 active-p how-active)
...                               ...
)                                )

                      Apply CROSSOVER
                      (exchange after 3rd thread).

Superthread of ML                Superthread in information reservoir
(                                (
(1 active-p how-active)           (#1 active-p how-active)
(2 active-p how-active)           (#2 active-p how-active)
(#3 active-p how-active)          (3 active-p how-active)
...                               ...
)                                )
```

The total procedure of our simulation is written as

```
PSEUDO CODE

     Initialize superthreads of ML and information reservoir

     while ( until the time end ) {
        for each c(i) {
            calculate the traffic
            sum up the total traffic
        }
log the total traffic
        mutation  at a given  mutation rate to superthread
        crossover at a given crossover rate to superthread
     }
```

When we use both crossovers and mutations, we find $1/f$ fluctuation in the lowest frequency region at a small crossover rate. See Fig. 2 for the PSD. One PSD with a small crossover rate(1 %) shows almost $1/f$ dependence in the lowest frequency of PSD but another with a little larger rate(16 %) shows that the PSD saturates to the white-noise.

When the crossover rate is large, exchanges between ML and information reservoir happen quite often. Since the exchange is carried out among many randomly generated superthreads, the simulation in this case is the same as a set of random processes.

When the crossover rate is small $\leq 1\%$, we find $1/f$ PSD as $f \to 0$. At a little larger crossover rate $\sim 10\%$, we find that $1/f$ PSD saturates to the white noise in the lowest frequency region. On the other hand, the small crossover rate implies that initial combination of threads in ML superthread changes little though the combination may be patched slightly by superthreads in information reservoir. The combination in ML superthread is modified so little that the system memorizes the initial configuration of ML superthread. In other words the system is not completely shuffled among the initial superthreads. $1/f$ PSD may disappear to lead to the white noise when the whole superthreads mixes with each other in the larger time scale.

$1/f$ behavior in the PSD is also characterized by the distribution function $P(T)$ of time staying in a state[1]. We define the distribution function $P(T)$ as the distribution of the time pausing in states where T is a period of almost

regular traffic between bursts in time series of the traffic. We determine that a burst is a period when the traffic is over some threshold value, for example 75% of the maximum of the traffic. This state is usually short. We estimate the pausing time distribution as T^{-2} (Fig. 3). Fig. 3 shows the cumulative distribution $\int_{t>T} P(t)dt$ is $\sim T^{-1}$. The relation between T^{-2} and $1/f$ is explained by a renewal theory[19] , which is a kind of semi Markovian process. This relation is known in chaos theories[1, 9]. The relation between our model or our observation and chaos is ambiguous, but our model suggests that we could understand our result in the framework of Non Markovian processes.

6 Summary and Discussion

In summary, we find $1/f$ fluctuations in Internet communication. We propose a model on a Mailing List to clarify that incomplete shuffling of the system information leads to $1/f$ fluctuations. We show that we can characterize features of $1/f$ PSD in the same way as chaos theories in conservative systems[1]. As a metaphor and an analogy, in the limit of small mutation and crossover rates, a fundamental dynamics of the system is a circulation over finite information, in one sense we may be able to say reversible, but lacks the properties little by little to be irreversible. It is important on $f^{-\mu}$ properties that the dissipation of the information must be a very small and intermediate between a random process and a deterministic one.

We know one similar example of the expressway traffic which shows $1/f$ fluctuations. Musha et. al. observed the car traffic (not traffic jam) [6]. For this example, they gave an explanation based on Burgers equation but it does not clarify the essence of $1/f$ fluctuations. Another example in the same field is fractal structures on IP Networking[7] and mainly packets streams(TCP/IP). Our time scale is very different from their one, so ours is not directly related with them.

Lastly we give a few comments on our use of GA. The usual GA has three procedures, mutation, crossover and selection. In addition to the procedures of mutation and crossover which we used, it requires fitness measure to determine the adjustment of optimization. Since our objective is not an optimization, we do not require the selection and evaluation function. In this point our procedure is essentially different from the GA. We utilize GA only for a process to exchange information at a very slow rate.

References

[1] K. Fukamachi, Europhys. Lett, **26**, 449(1994).

[2] M. B. Weissman, Rev. Mod. Phys. **60**, 537 (1988).

[3] T. Musha, A. Nakajima, and H. Akabane, Jpn. J. Appl. Phys. **27**, L311 (1988).

[4] T. Musha, B. Gabor, and M. Shoji, Phys. Rev. Lett. **64**, 2394 (1990).

[5] R. F. Voss, Phys. Rev. Lett. **68**, 3805 (1992).

[6] T. Musha and H. Higuchi, Jpn. J. Appl. Phys. **15**, 1271 (1976).

[7] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson, IEEE/ACM Trans. on Networking, Vol.2, No. 1, 1-15(1994)

[8] Y. Pomeau and P. Manneville, Comm. Math. Phys. **74**, 189 (1980).

[9] Y. Aizawa and T. Kohyama, Prog. Theor. Phys. **71**, 847 (1984).

[10] D.E.Comver *Internetworking With TCP/IP, Principles, Protocols, and Architecture*, (Prentice Hall, Englewood Cliffs, NJ, 1991).
W. R. Stevens, *UNIX Network Programing*, (A Simon & Schuster Company, NJ, 1990),

[11] Costales, Allman, and Rickert, *Sendmail* (O'Reilly & Associates).
One of the implementation of the Mail system is "Sendmail Release 8" from U.C.Berkeley. See also the following drafts on more technical points.
rfc821., "SIMPLE MAIL TRANSFER PROTOCOL", 1982.
rfc822, "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", 1982.
rfc1425, "SMTP Service Extensions", 1993
rfc1426, "SMTP Service Extension for 8bit-MIMEtransport", 1993
rfc1427, "SMTP Service Extension for Message Size Declaration", 1993
rfc1428, "Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME", 1993.

[12] E. Korl, *The Whole Internet: Guide for the beginners* (Addison-Wesley, 1993)

[13] One example of the implementation of a Mailing List Server is my freeware(under GNU Public License). It is contributed to fj.sources as
<FUKACHAN.94Sep12063149@axion.phys.titech.ac.jp>
<FUKACHAN.94Sep12063412@axion.phys.titech.ac.jp>
<FUKACHAN.94Sep12063554@axion.phys.titech.ac.jp>
<FUKACHAN.94Sep12063737@axion.phys.titech.ac.jp>
You can obtain it by anonymous ftp "ftp://ftp.phys.titech.ac.jp/pub/net/fml-current" and by WWW "http://www.phys.titech.ac.jp/uja/fml/".

[14] *Genetic Algorithm*, edited by H. Kitano (Sangyou Tosho, Tokyo, 1992)[in Japanese].

[15] rfc1594, "FYI on Questions and Answers Answers to Commonly asked "New Internet User" Questions", 1994.

[16] S. J. Leffler, M. K. McKusick, M. J. Karels and J. S. Quarterman *The design and implementation of the 4.3BSD UNIX Operating System*, (Addison-Wesley, 1989).

[17] Guy L. Steele Jr. *Common Lisp* (Digital Equipment Corporation, Bedford, 1990)

[18] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, 1974) p. 252.

[19] W.Feller, *Introduction to Probability Theory*

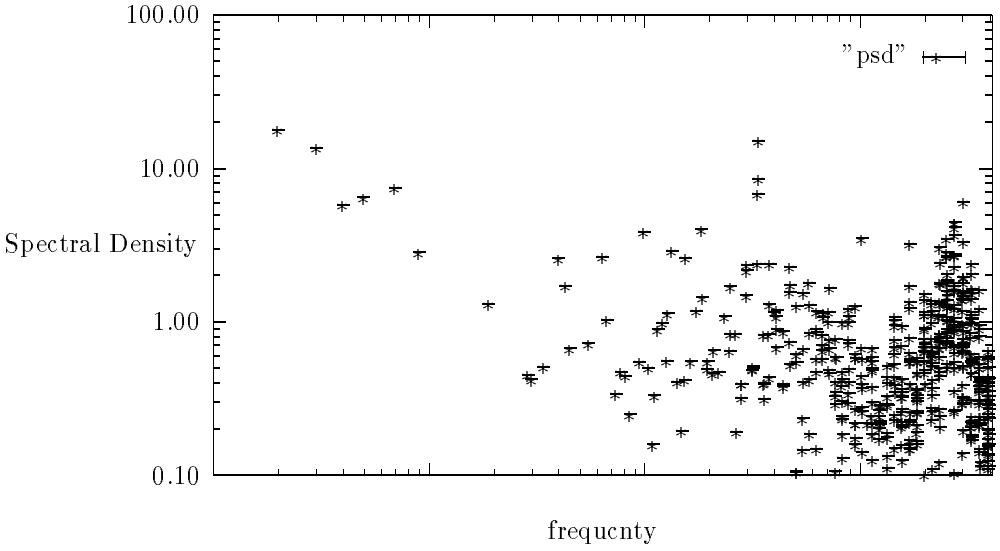

Figure 1: PSD calculated from data of a Mailing List log. It is plausible that the long time correlation appears in the scale over one week, since the activity in the scale less than a week must work almost periodically and randomly.

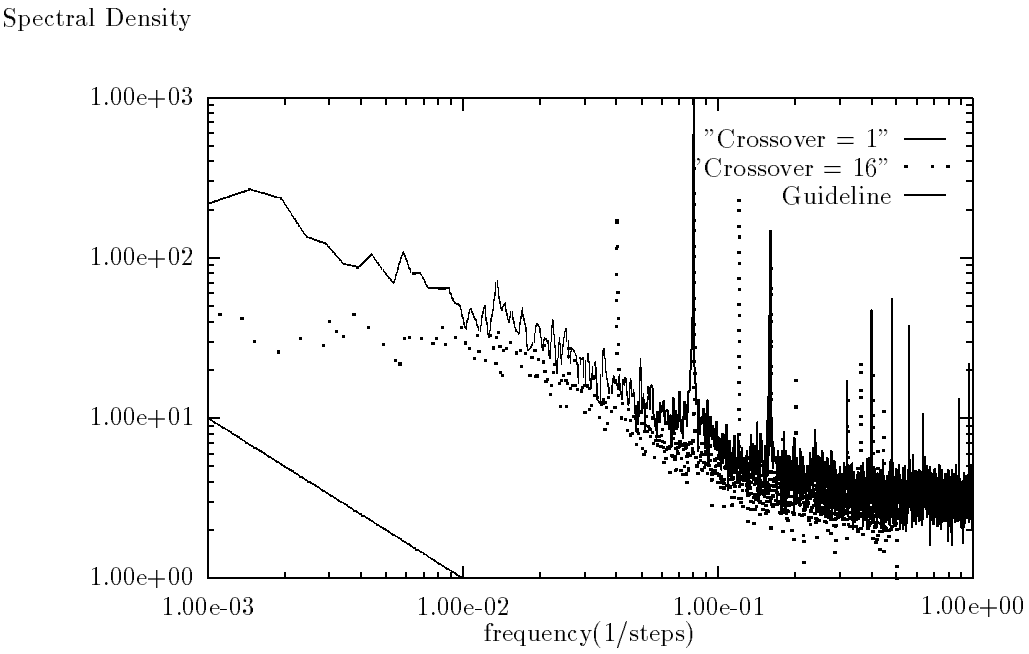

Figure 2: PSD of our model shows. The straight line is a guide for the eye representing $1/f$-dependence. The full lines are crossover rate = 1 % and the dotted lines are crossover rate = 16 %. The dotted ones saturate to the white noise in the low frequency but the other with a small crossover rate has a $1/f$ form. One superthread consists of 10 threads (subjects). There are 10 superthreads in information reservoir. Both cases(1% and 16%) have the same mutation rate 0.03.

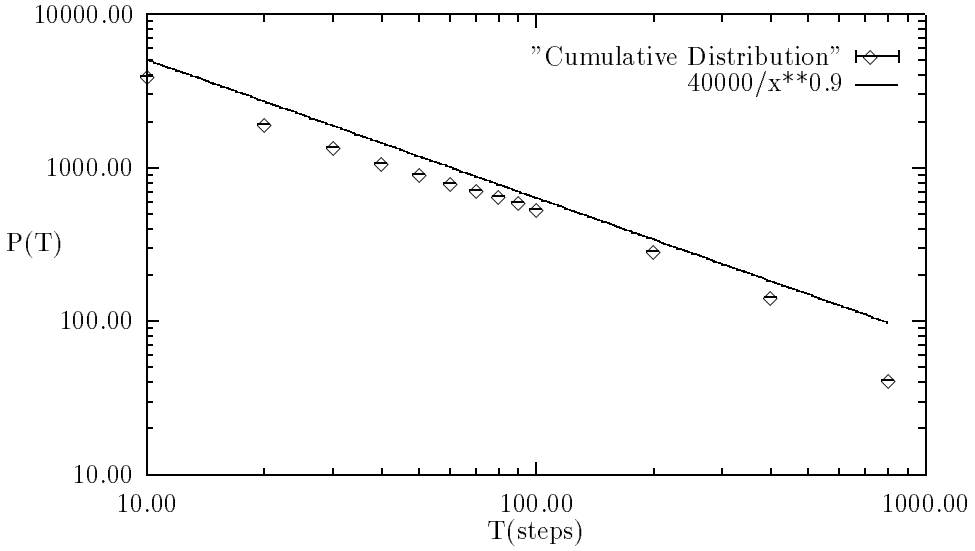

Figure 3: Cumulative distribution of the pausing time distribution. The straight line is a guide for the eye representing $T^{-0.9}$. T^{-1} here corresponds to T^{-2} in raw pausing time distribution. The distribution function $P(T)$ of time T which is a period between bursts in time sequences of the traffic. In this case we determine that a burst is a period when the traffic is over some threshold value, 75 %, of the maximum of the traffic.