



Helsinki University of Technology
Department of Communications and Networking
S-38.3138 Networking Technology, Special Assignment

Effect of Service Time Distribution on the Performance of P2P File Sharing

Made by: Manoj Bhusal
80545E
mbhusal@cc.hut.fi
Supervisor: Samuli Aalto
Completed: September 09, 2008

Abstract

Few studies have been done on the performance analysis of peer to peer (P2P) file sharing systems. They are limited to modelling the systems where the service time is exponentially distributed. In this special assignment, the effect of other service time distributions such as Erlang-2 and hyper-exponential-2 distribution in addition to the exponential distribution on the performance P2P system is studied. The different service time distributions model the kind of service the peers experience in a particular P2P system. A tandem queue consisting of two queues is considered to model the system under study. Although most of the P2P applications employ multiple chunks in file sharing, one chunk model is considered here for its simplicity. This model is first analyzed using deterministic fluid model. A more detailed analysis is done using the Markov chain analysis. The study also takes into account the stability of the system in relation to the arrival and departure rates of the peers. The simulations are performed finally which verify the analytical results.

Contents

1	Introduction	1
2	System Description	2
2.1	Markovian Analysis	3
2.2	Stability Consideration:	5
3	Description of Simulator	6
3.1	Probability Distributions	7
3.2	Generation of random variables	9
4	Results and Discussion	10
4.1	Effect of Arrival the Rate of Downloaders	10
4.2	Effect of Departure Rate of Seeds	12
4.3	Stability Consideration	14
4.4	Comparison of Simulation and Numerical Results	16
5	Conclusions	17
	References	19
	Appendix A. Markovian Analysis in <i>Mathematica</i>:	20
	Appendix B. Simulation of the P2P system in <i>Mathematica</i>:	22

1 Introduction

P2P file sharing systems and other applications using P2P mechanism for services are producing significant volume of Internet traffic. Studies and measurements have shown that P2P traffic in the Internet accounts for 40% to 80% depending upon the region of observation. This has been due to the rapid development of P2P systems in the last decade. Although Napster, Gnutella and Kazaa were the popular systems during early 2000, eDonkey, BitTorrent and its variants have attracted large number of users in recent years [1][2].

With the increase in bandwidth accessible to the user, P2P systems' uses are no longer limited to the sharing of delay-tolerant files. Recently many multimedia streaming web applications such as Youtube and JumpTV have been successful in attracting huge amount of users. These systems are based on the client/server architecture where the servers unicast the content to the users of the service. Thus, they do not scale well as the number of users increases and require server replication to get around the problem of scalability. This has motivated the application of P2P technology for the delivery of real time content such as streaming video, audio files and video on demand which is evident in few of the already developed systems such as Sopcast [3].

Although the implementation of protocol may vary from one to another, all the P2P systems have the inherent property of allowing the users to obtain and distribute a resource (file) in a cooperative manner. For using the service of the system, each participating peer contributes to the system. In traditional file sharing applications where a client requests a file from a server (eg: FTP server), an increasing number of requests from new clients deteriorates the service that can be dispatched by the server. However, in a P2P system, more users joining the system means more resources being available to the other participating peers and hence contributing to the scalability of the system.

The other main difference between the client/server system and P2P system is that while a client downloads from the same server in the client/server system, a peer may be downloading from any other peer which is present in the system at that time. The presence of a peer in the system is not permanent. The peer may leave the system while the other peer is being served by it. In that case, the downloading peer seeks to download from the other peers which have the resource it was downloading from the peer which just left the system. But in a client/server system, the server is assumed to be present permanently and if somehow the sever is not reachable, the service is unavailable.

File sharing protocols such as used by BitTorrent divide a file to be shared into multiple parts known as chunks. The peers download the chunks which they do not have yet and upload the chunks they already have with other peers [4]. Considerable work has been done in understanding system design and traffic measurement of popular P2P systems such as in [5]. Effort has been going on now in studying the performance issues such as the download time, file availability, lifetime of the file sharing process and the performance of P2P system with multiple chunks [6][7][8].

The performance parameters of a P2P system, among other things, are affected by the distribution of the arriving customers, the service time distribution and the lifetime of the seeds. Understanding the effect of these parameters on performance is essential in designing the system to take into account these effects. In this special assignment, the goal is to study the performance of a P2P system for a given service

time distribution. The work in this report primarily includes,

- Development of a Markov model for one chunk P2P file-sharing network.
- Simulation of the Markov model.
- Comparison of the results of simulation and the numerical analysis of the P2P system.

The report is organized as follows: section 2 describes the model that is going to be used for simulations, the numerical analysis of the model using Markovian analysis and a brief discussion on the stability of the system using a deterministic fluid model. The construction of the simulator is described in section 3. The results of the simulations are discussed in section 4 and also a brief comparison of numerical analysis and results of the simulation is carried out. Finally, summary of the results and conclusion is presented.

2 System Description

The model used for study is a simple tandem queue as shown in Fig. 1. Although, the protocols such as BitTorrent use multiple chunks for file sharing, the single chunk model has been considered for its simplicity in the analysis which can still show some important characteristics of the system.

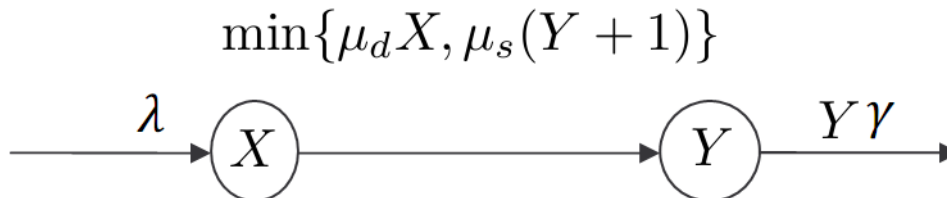


Figure 1: A model for file sharing(single chunk)

A P2P system consists of three kinds of peers participating in the file sharing process namely downloaders, leechers and seeds. Downloaders are the newly arriving peers which have no chunks with them yet. The leechers are the peers which already have one or more chunks of the file being shared but do not have a complete copy of the file with them yet. Finally, seeds are the peers which already have a full copy of the file being shared.

Since the system being studied is based on the one chunk model, there are only downloaders and seeds but no leechers in the system. When a downloader has completed the downloading of a file, it becomes a seed. Furthermore, we assume that there is one particular seed which never leaves the system. Although the provision of a permanent seed is not present in BitTorrent like systems, this can be implemented by introducing some amount of centralized control so that the perpetuity of the file sharing process is ensured. Otherwise, the file sharing process fails as soon as there are no seeds at any moment of the file sharing process.

New peers seeking a given file arrive to the system according to a Poisson process with rate λ . The download rate is denoted by μ_d and upload rate by μ_s . We assume

that all the peers have the same uploading and downloading capacity, *i.e.* $\mu_d = \mu_s = \mu$. After the downloaders arrive to the system, they may immediately start downloading if there are any vacant seeds or wait till any seed gets vacant. So the total download time perceived by the downloader is the sum of waiting time in the queue and the time it takes to actually download a file. From here on, the sojourn time is referred to as download time.

The seeds leave the system randomly with rate γ . The time that a seed spends in the system is also exponentially distributed with parameter γ . This will have major effect in the performance of the system since for larger γ there will be small number of seeds at any given time and thus downloaders have to wait for longer time in the queue before they get their service. The rate γ is influenced by the incentive the seeds get after downloading a given file to remain in the system. It is assumed that γ is constant during the whole process.

The P2P system starts with a single seed which has the file that other peers are interested in downloading. At any time t , let $X(t)$ be the random variable denoting the number of downloaders and $Y(t)$ be the random variable denoting the number of seeds excluding the permanent seed. Then, the total download capacity of the downloaders is $\mu_d X(t)$ and the upload capacity of the seeds is $\mu_s(Y(t) + 1)$. Upon arrival of a downloader to the system, if $\mu_d X(t) \leq \mu_s(Y(t) + 1)$, then it can start downloading immediately. However, if $\mu_d X(t) > \mu_s(Y(t) + 1)$, then the downloader has to wait in the queue until other downloaders ahead of it in the queue complete the downloading of the file. This means the total service rate of the system at time t is $\min\{\mu_d X(t), \mu_s(Y(t) + 1)\}$. Since $\mu_d = \mu_s = \mu$, the service rate of the system is $\min\{\mu X(t), \mu(Y(t) + 1)\}$. The Markovian analysis is presented below for the special case of upload and download times being exponentially distributed by constructing a continuous time Markov chain.

2.1 Markovian Analysis

The state of the system is given by the pair (X, Y) , and $\pi_{i,j}$, where $0 \leq i < \infty$ and $0 \leq j < \infty$, are the state probabilities. The state diagram is given in Fig. 2. The method of global balance is used to calculate the steady state probabilities. Since the state space extends from $(0, 0)$ to (∞, ∞) , the solution involving all the states did not seem to be tractable, so truncation of the state space is done such that the truncated states have negligible steady state probabilities. By truncating the state space such that X can extend from 0 to m and Y can extend from 0 to n , we can write the following global balance equations.

probability flow into the state=probability flow out of the state

For all i such that $0 < i < m$,

$$\begin{aligned} \pi_{i-1,0}\lambda + \pi_{i,1}\gamma &= \pi_{i,0}(\lambda + \mu) \\ \pi_{i-1,n}\lambda + \pi_{i+1,n-1} \min\{i+1, n\}\mu &= \pi_{i,n}(\lambda + n\gamma) \end{aligned}$$

For all j such that $0 < j < n$,

$$\begin{aligned} \pi_{1,j-1}\mu + \pi_{0,j+1}(j+1)\gamma &= \pi_{0,j}(\lambda + j\gamma) \\ \pi_{m-1,j}\lambda + \pi_{m,j+1}(j+1)\gamma &= \pi_{m,j} \min\{m, j+1\}\mu + \pi_{m,j}j\gamma \end{aligned}$$

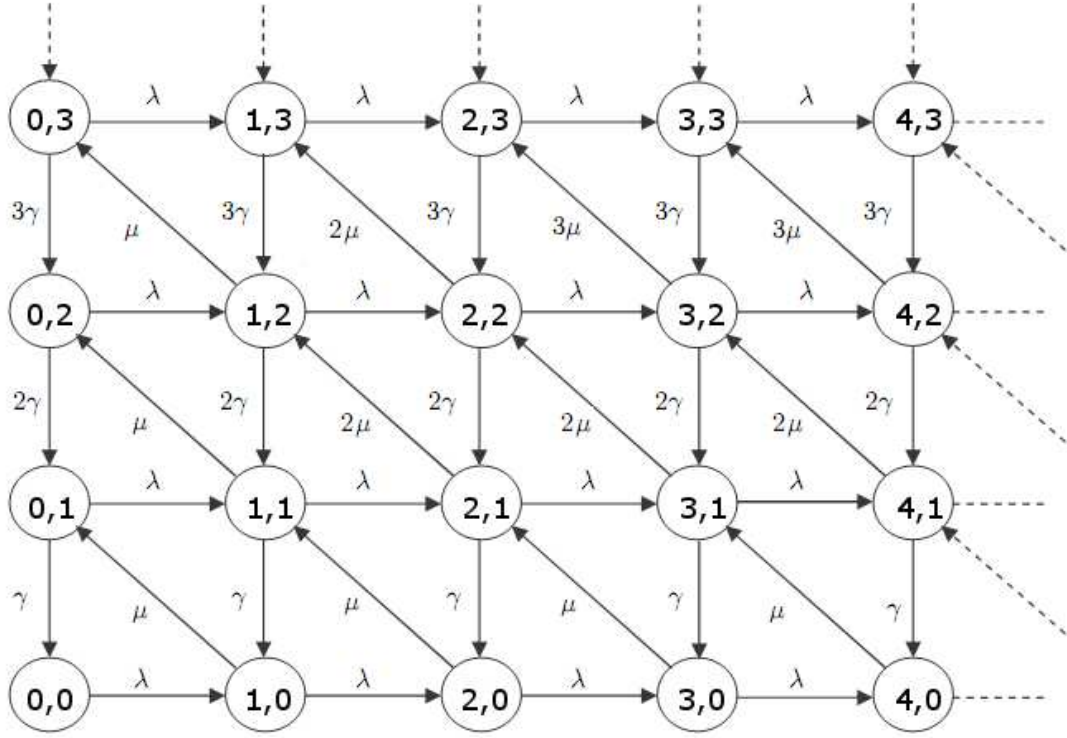


Figure 2: state diagram for Markovian analysis of file sharing.

For all i such that $0 < i < m$ and j such that $0 < j < n$

$$\pi_{i-1,j}\lambda + \pi_{i+1,j-1} \min\{i+1, j\}\mu + \pi_{i,j+1}(j+1)\gamma = \pi_{i,j}(\lambda + \min\{i, j+1\}\mu + j\gamma)$$

For the states in the corner of the state diagram,

$$\begin{aligned} \pi_{0,0}\lambda &= \pi_{0,1}\gamma \\ \pi_{m-1,0}\lambda + \pi_{m,1}\gamma &= \pi_{m,0}\mu \\ \pi_{1,n-1}\mu &= \pi_{0,n}(\lambda + n\gamma) \\ \pi_{m,n}\lambda &= \pi_{m-1,n}n\gamma \end{aligned} \tag{1}$$

Using the normalization condition,

$$\pi_{0,0} + \pi_{0,1} + \dots + \pi_{0,n} + \pi_{1,0} + \pi_{2,0} + \dots + \pi_{m,0} + \pi_{1,1} + \pi_{1,2} + \dots + \pi_{m,n} = 1 \tag{2}$$

The solution obtained for the steady state probabilities using the $(m+1) \times (n+1)$ equations in (1) and (2) are used in calculating the mean, standard deviation and coefficient of variation of the different parameters of the system. Subsequently, these values are used in calculating the download time and the lifetime of seeds and the respective standard deviations and coefficients of variation using the Little's result as given in (3). These values are then compared with results of the simulation

Little's Result: gives a simple relationship between the mean number of customers $E(N)$, mean sojourn time $E(S)$ and λ the average arrival rate of the customers in a stable system.

$$E(N) = \lambda E(S) \tag{3}$$

2.2 Stability Consideration:

Using the deterministic fluid model as studied in [9], the stability bounds can be derived. The following quantities are used to describe the fluid model,

- $x(t)$ number of downloaders in the system at time t .
- $y(t)$ number of seeds in the system at time t excluding the permanent seed.
- λ , μ and γ are the parameters of the system.

Then the deterministic fluid model for the evolution of the number of downloaders and seeds is given by,

$$\begin{aligned}\frac{dx(t)}{dt} &= \lambda - \min\{\mu x(t), \mu(y(t) + 1)\}, \\ \frac{dy(t)}{dt} &= \min\{\mu x(t), \mu(y(t) + 1)\} - \gamma y(t)\end{aligned}\tag{4}$$

where $x(0) = 0$ and $y(0) = 0$ implying that at time $t = 0$, there is only one seed which is permanent and no downloaders. Let \bar{x} and \bar{y} be the equilibrium values of $x(t)$ and $y(t)$. The solution for the differential equations (4) exists and the system is stable for the following cases:

- $\gamma < \mu$ and $\lambda < \infty$
- $\gamma \geq \mu$ and $\lambda < \frac{1}{\frac{1}{\mu} - \frac{1}{\gamma}}$

The steady state values of \bar{x} and \bar{y} for the above cases are

$$\bar{x} = \frac{\lambda}{\mu} \text{ and } \bar{y} = \frac{\lambda}{\gamma}.$$

If $\gamma \geq \mu$ and $\lambda \geq 1/(\frac{1}{\mu} - \frac{1}{\gamma})$, then $x(t)$ increases without limits as $t \rightarrow \infty$, but $y(t)$ still has a limiting value,

$$\bar{x} = \infty \text{ and } \bar{y} = \frac{\mu}{\gamma - \mu}.$$

Fig. 3 shows the evolution of the number of downloaders and seeds for the case $\mu \geq \gamma$

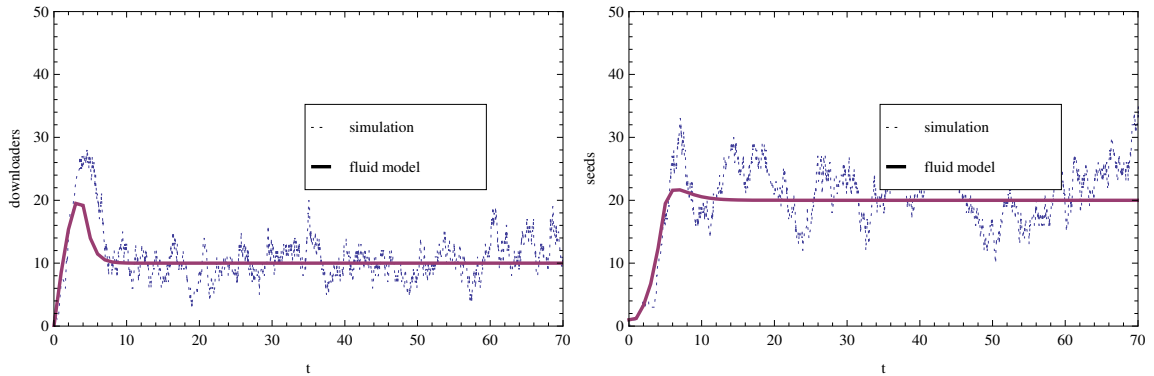


Figure 3: The number of downloaders (left) and seeds (right) as a function of time. $\lambda/\mu = 10$, $\lambda/\gamma = 20$, $\mu \geq \gamma$

so that the system is stable for any value of λ . In the beginning, the number of seeds is not sufficient to serve the incoming download requests. This can be seen from the large increase of downloaders. As the time progresses, the downloaders change into

seeds such that there are sufficient number of seeds to serve all the incoming download requests and the system stabilizes.

Fig. 4 shows the evolution of the system for $\lambda = 10$ and different values of μ and γ . When $\gamma = 0.5$ and $\mu = 1$ so that $\gamma < \mu$, new seeds arise faster than existing seeds leave the system. The number of seeds quickly overtakes the number of downloaders in the system and both quantities reach their steady state values. When $\gamma = \mu = 1$, the system is still stable but it takes more time for the number of seeds and downloaders to reach their steady state values. When $\gamma = 2$ and $\mu = 1$ so that $\gamma > \mu$ and $\lambda > \frac{1}{\frac{1}{\mu} - \frac{1}{\gamma}}$, the seeds leave the system faster than the new seeds arise in the system. The number of seeds in the system is never sufficient to serve the incoming download requests and the system becomes unstable. Once the system becomes unstable, the the number of seeds goes on decreasing from the stable steady state value of λ/γ . As λ takes

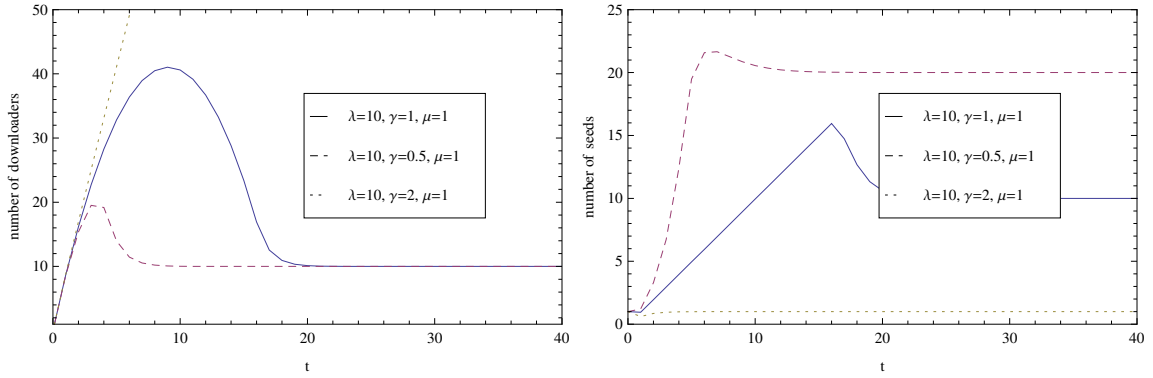


Figure 4: The number of downloaders (left) and seeds (right) as a function of time.

values larger than the stability limit $1/(1/\mu - 1/\gamma)$, the number of seeds stabilizes to $\bar{y} = \mu/(\gamma - \mu)$, *i.e.* 1 in this case.

3 Description of Simulator

The Markov model presented in section 2 is simulated in *Mathematica*. *Mathematica* can be used as an application as well as a programming language. Due to the available collection of rich built-in functions and data structures, complex tasks can be solved with relative ease in *Mathematica*. The use of already available functions such as random number generator and flexible lists to build the simulator make the task simple which would have been tedious in any other programming language such as C or C++ or even in Java.

To develop the simulator for the model, an open queuing network consisting of two queues in tandem is constructed. The new downloaders arriving to the system according to a Poisson process with rate λ enter the first queue whose servers are the seeds already present in the system. Since the arrivals follow a Poisson process, the inter-arrival time is exponentially distributed. After the arrival of a downloader, the required time for its service (downloading of the file) is drawn from a distribution which may be exponential, Erlang- k or hyper-exponential. The next arrival time of a downloader is the new inter-arrival time added to the current time.

Since the upload and download capacities are assumed to be equal in the model, the number of seeds and downloaders present in the queues determine whether a newly

arriving downloader has to wait or can immediately start downloading. If the number of downloaders is greater than the number of seeds, the downloader has to wait in the queue before a seed becomes available for uploading. The downloader leaves the first queue after its service time expires (completion of file download) and enters the second queue. The downloader leaving the first queue is equivalent to the arrival of new customer (seed) in the second queue. It also has the effect of increasing the number of servers (seeds uploading to the downloaders) in the first queue.

When the seed enters the second queue its service time (the lifetime of seed in the system), which is exponentially distributed with parameter γ , starts. After a downloader becomes a seed, the downloaders' queue is checked to see if there are any downloaders waiting for service. In case there are any waiting downloaders, the new seed starts serving (uploading) the waiting downloader. The next departure time of the downloader from the first queue is the minimum of the remaining service times of all the downloaders being served by the seeds.

When the service time of a seed (lifetime) in the second queue expires, it leaves the system and the time to the next departure of a seed is calculated. The new departure time is the minimum of the remaining service times of the seeds in the second queue added to current time. The leaving seed may or may not be serving a downloader before its departure. If it was uploading to a downloader, the new remaining service time of the downloaders in the first queue is calculated. The downloader whose service gets interrupted is placed ahead of all the waiting downloaders so that it can immediately resume downloading when a seed becomes free.

The time to the next event is the minimum of the new remaining service time of downloaders, the time to the next arrival of a downloader and the time to the next departure of a seed. The generation of the random variables of different distributions is described in section 3.2 which follows from the properties of these distributions as discussed in section 3.1.

Simulations are done for different values of the arrival rate of downloaders, λ , keeping the download rate, μ , and the departure rate of the seeds, γ , constant. Similarly, simulation runs are made for different values γ keeping λ and μ constant. These simulations are repeated for the cases where the service time of downloaders is distributed according to Erlang- k distribution and hyper-exponential distribution.

3.1 Probability Distributions

Poisson Distribution: A random variable X is Poisson distributed with parameter λ if its probability mass function is given by

$$P(X = n) = \frac{\lambda^n}{n!} e^{-\lambda}, \quad n = 0, 1, 2, \dots$$

For the Poisson distribution,

$$E(X) = \sigma^2(X) = \lambda$$

$$C_X = \frac{\sigma(X)}{E(X)} = \frac{1}{\sqrt{\lambda}}$$

where $E(X)$ is the mean, $\sigma^2(X)$ is the variance of X and C_X is the coefficient of variation.

Exponential distribution: A random variable X is exponentially distributed with parameter μ if its probability density function is given by

$$f_X(x) = P(X \in dx) = \mu e^{-\mu x}, \quad x \geq 0$$

For this distribution,

$$E(X) = \frac{1}{\mu}, \quad \sigma^2(X) = \frac{1}{\mu^2}$$

$$C_X = \frac{\sigma(X)}{E(X)} = 1$$

The exponential random variable has memoryless property. It means if X is time taken for certain event to happen which is an exponentially distributed with parameter μ and if it has lasted t seconds, the remaining time for the event to take place is still exponentially distributed with same parameter μ . Mathematically,

$$P(X > t + x | X > t) = P(X > x) = e^{-\mu x}$$

Erlang- k distribution: A random variable X is Erlang- k distributed if it is the sum of k independent identically distributed random variables X_1, X_2, \dots, X_k [10]. If μ denotes the parameter of each of the exponential random variable, the probability density function is given by

$$f_X(x) = P(X \in dx) = \mu \frac{\mu x^{k-1}}{(k-1)!} e^{-\mu x}, \quad x > 0.$$

It is a phase type distribution. The phase diagram of Erlang- k distribution is shown in Fig. 5. where each of the phases has parameter μ . For this distribution,



Figure 5: Phase diagram of the Erlang- k distribution

$$E(X) = \frac{k}{\mu}, \quad \sigma^2(X) = \frac{k}{\mu^2}$$

$$C_X = \frac{\sigma(X)}{E(X)} = \frac{1}{\sqrt{k}} < 1$$

For $k = 2$, we have

$$C_X = \frac{1}{\sqrt{2}} = 0.707$$

Hyper-exponential distribution A random variable X is hyper-exponentially distributed if X is X_i with probability $p_i, i = 1, \dots, k$ where each of the X_i is exponentially distributed with parameter μ_i [10]. It is also a kind of phase type distribution. The phase diagram of hyper-exponential distribution is shown in Fig. 6. The probability density function is given by

$$f_X(x) = P(X \in dx) = \sum_{i=1}^k \mu_i e^{-\mu_i x}, \quad x > 0.$$

For this distribution,

$$E(X) = \sum_{i=1}^k \frac{p_i}{\mu_i}, \quad \sigma^2(X) = 2 \sum_{i=1}^k \frac{p_i}{\mu_i^2} - E(X)^2$$

$$C_X = \frac{\sigma(X)}{E(X)} = \sqrt{\frac{2 \sum_{i=1}^k \frac{p_i}{\mu_i^2}}{(\sum_{i=1}^k \frac{p_i}{\mu_i})^2} - 1} \geq 1$$

For $k = 2$, $p_1 = 0.3$, $p_2 = 0.7$, $\mu_1 = 0.6$ and $\mu_2 = 1.4$, we have

$$C_X = 1.18$$

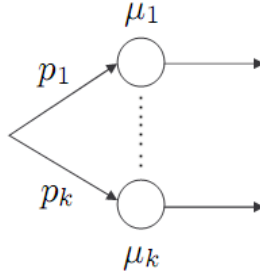


Figure 6: Phase diagram of hyper-exponential distribution

3.2 Generation of random variables

Exponential random variable: This method of random variable generation involves inverting the cumulative distribution function $F(x) = P(X \leq x)$ associated with random variable X . Since $0 \leq F(x) \leq 1$, by generating random numbers uniformly distributed over $(0, 1)$, a sample of random variable X can be produced by inverting its cumulative distribution function. Thus, to generate random variable X with cumulative distribution function $F(x)$, we set $U = F(x)$ and obtain

$$X = F^{-1}(U)$$

where U is uniformly distributed in $(0, 1)$.

For exponential distribution with parameter μ , we have

$$F(x) = P(X \leq x) = 1 - e^{-\lambda x}, \quad x \geq 0$$

Solving for X in $U = F(X)$ gives

$$X = F^{-1}(U) = -(1/\mu) \log(1 - U)$$

Erlang- k random variable: Since the Erlang- k distribution is equivalent to the sum of k independent and identically distributed exponential random variables, we can obtain an Erlang- k random variable by adding k exponential random variables.

Hyper-exponential random variable: A hyper-exponential random variable takes the value $X = X_{\mu_i}$ with probability p_i , where $X = X_{\mu_i}$ are the k exponential random variables each with rate μ_i . To generate a hyper-exponential random variable, first a random number uniformly distributed in $(0, 1)$ is generated. This random number is used to choose the parameter μ_i of phase corresponding to the point probability p_i . Then the selected parameter is used to generate the exponential random variable as discussed above which gives the required sample of the hyper-exponential random variable. The parameters p_i and μ_i can be chosen so as to have the desired mean of the distribution.

4 Results and Discussion

The results of the simulations with regard to the mean values of the performance parameters and their variability under different download and upload conditions are discussed here with graphical illustrations. The figures show the metrics comparing cases where the distribution of the service time of downloaders are exponential, Erlang-2 and hyper-exponential-2. Then the simulation and numerical results are compared. The parameters of the service time distributions of downloaders are chosen such that the mean value is 1. The parameter of the exponential distribution μ is set to 1 so that its mean and coefficient of variation are equal to 1. For Erlang-2 distribution, the parameter of each of the two phases is set to 2. Using the formulas presented in section 3.1, the mean of the distribution is 1 and coefficient of variation is 0.71. In case of hyper-exponential-2 distribution with two phases, the phases have exponential distribution with parameter $\mu_1 = 0.6$ and $\mu_2 = 1.4$ and their probabilities are $p_1 = 0.3$ and $p_2 = 0.7$ respectively so that mean is 1 and coefficient of variation is 1.18.

4.1 Effect of Arrival the Rate of Downloaders

Fig. 7 shows the mean and coefficient of variation of the download time and the lifetime

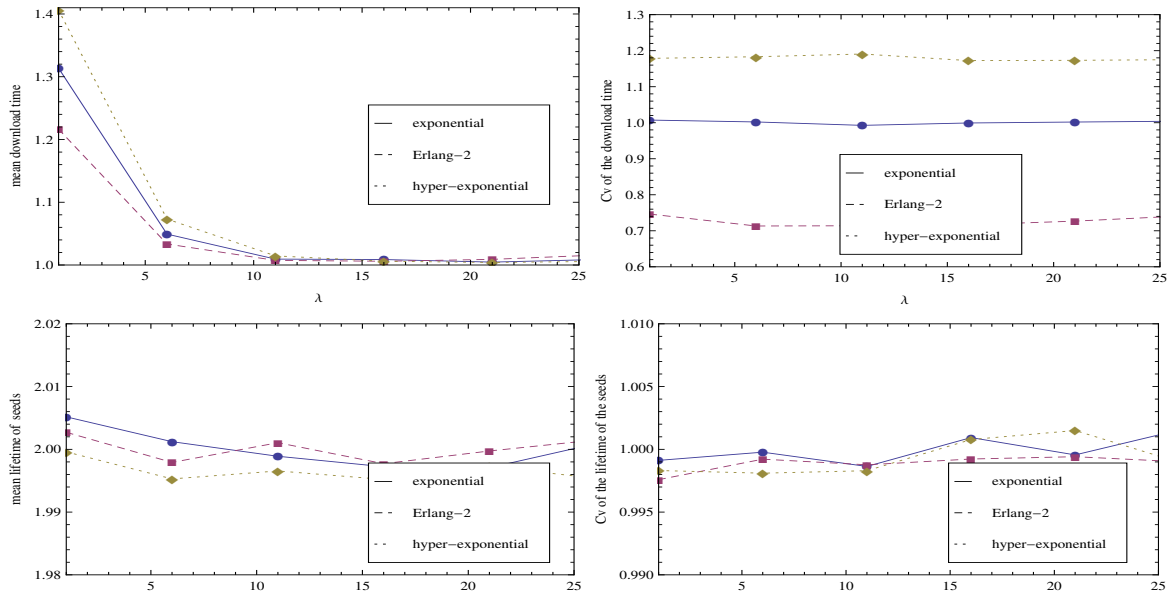


Figure 7: Mean and the coefficient of variation (C_v) of the download time and the lifetime of seeds as a function of λ . $\mu = 1$, $\gamma = 0.5$.

of seeds respectively as a function of peer arrival rate λ for the case $\mu = 1$, $\gamma = \frac{1}{2}$ so that $\gamma < \mu$. This guarantees stability for any $\lambda < \infty$. The different lines represent the cases for different distributions of service time namely exponential, Erlang-2 and hyper-exponential. It can be observed from the top left part that the mean download time decreases with the increase in the arrival rate. However, the download time increases with increase in λ for small values. This is not clear in this figure as it does not take into account the small scale values of λ around 1. The results for the small scale values of λ will be shown in the coming section which compares the numerical and simulation results. The decrease in download time for increasing λ is encouraging since the system performance becomes better as the peer arrival rate increases. Since the rate of seed departure is smaller than the service time of the downloaders, the number of seeds increases with increase in λ . This makes the downloaders queue is equivalent to $M/G/\infty$ for large values of λ .

The top right part of Fig. 7 shows that the coefficient of variation of the download time is constant for the different cases of service time distributions. For all the service time distributions, the coefficient of variation is approximately equal to the that of corresponding service time distribution. Thus it can be inferred that the distribution of the download time is similarly distributed to the service time of the downloaders. The lifetime of the seeds and its coefficient of variation is approximately 1. The distribution of download time does not affect the mean and coefficient of variation of the lifetime of seeds in the system as it should be.

Fig. 8 shows the mean and coefficient of variation of the number of downloaders and seeds. The number of downloaders increases with the increase in the arrival rate.

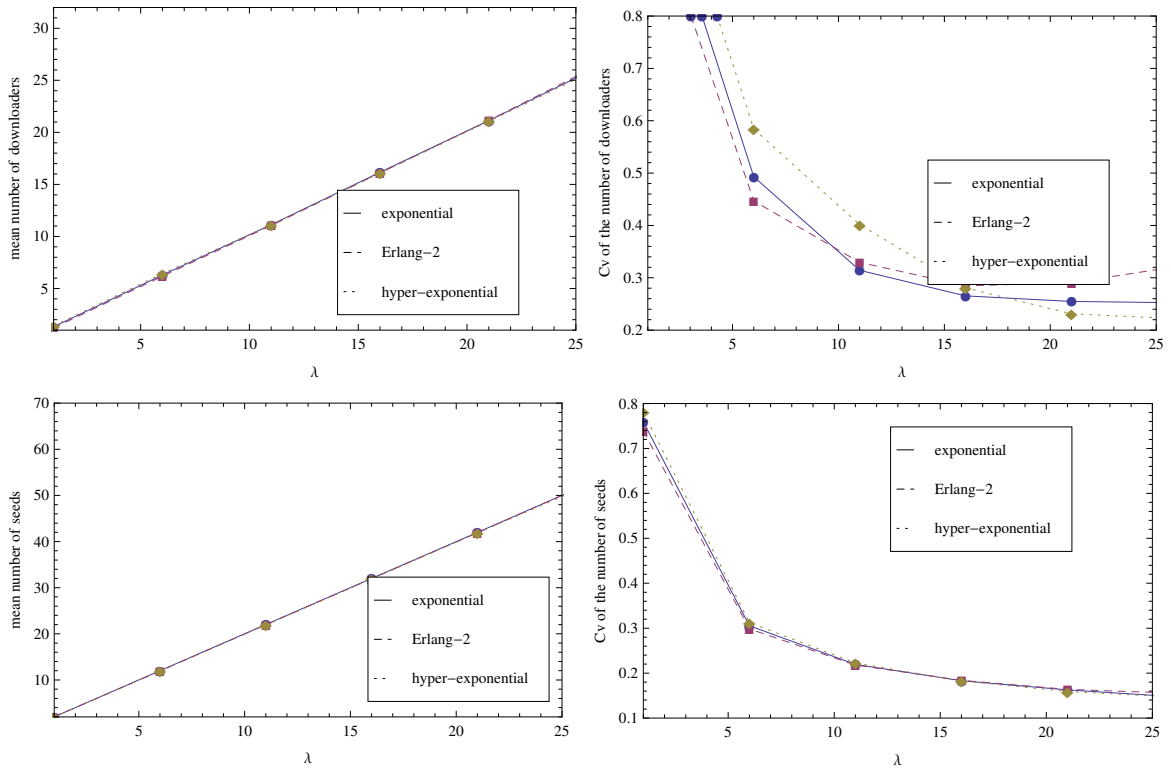


Figure 8: Mean and coefficient of variation (C_v) of downloaders and seeds as a function of λ . $\mu = 1$, $\gamma = 0.5$.

Moreover, it shows that the mean number of downloaders in the system is linearly

dependent on the arrival rate which is in line with the Little's result. The coefficient of variation of downloaders decreases for increasing λ and the value is approximately $1/\sqrt{\lambda}$. It implies that the downloaders' queue behaves as a $M/G/\infty$ for this stable system and thus queue length is Poisson distributed. In the bottom part of Fig. 8, it can be seen that the mean number of seeds increases linearly with λ which again is in line with the Little's result for the queue of seeds. The coefficient of variation of the number of seeds decreases for increasing λ . The trend shows that the distribution of the number of seeds is Poissonian with parameter λ/γ . So it can be inferred from here that the queue of seeds behaves as $M/M/\infty$ in this case.

4.2 Effect of Departure Rate of Seeds

The Fig. 9 shows the mean and coefficient of variation of the downloaders and seeds respectively as a function of $\frac{1}{\gamma}$ for the case $\mu = 1$, $\lambda = 1$ so that $\lambda < \mu$. This guarantees stability for any $\frac{1}{\gamma} > 0$. The mean download time decreases with the decrease in the

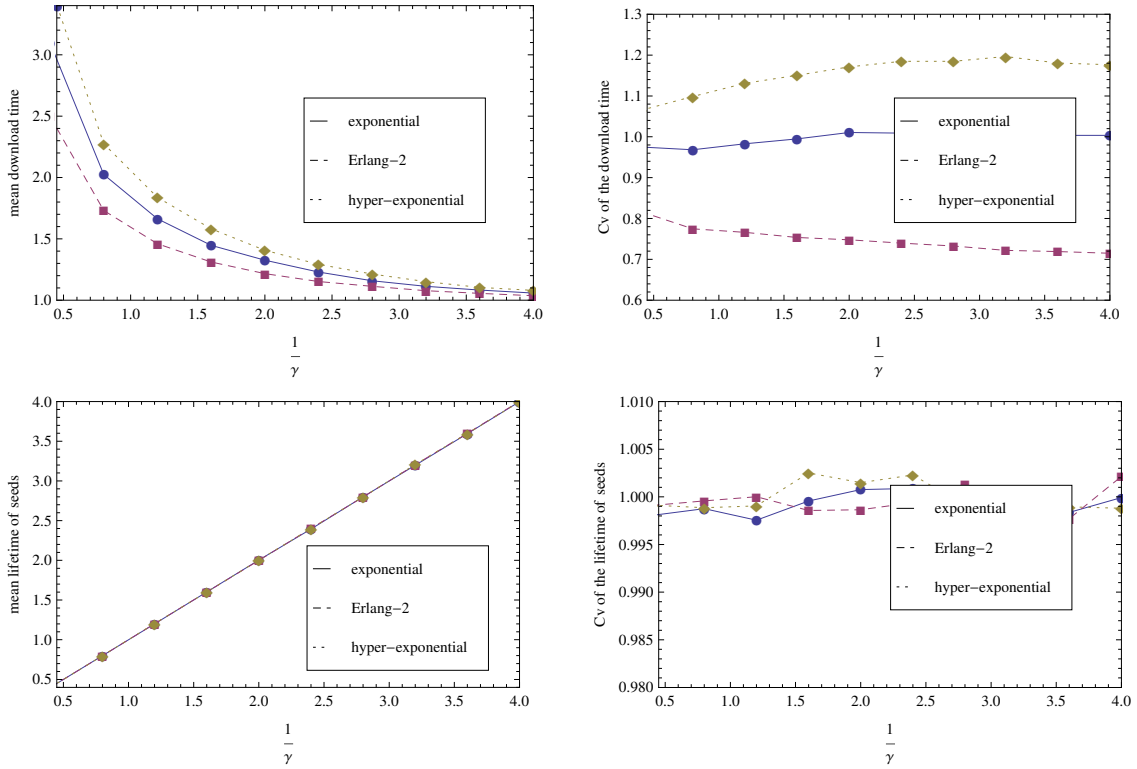


Figure 9: Mean and coefficient of variation (C_v) of the download time and lifetime of seeds as a function of $\frac{1}{\gamma}$. $\mu = 1$, $\lambda = 1$.

seed departure rate (γ). It stabilizes to 1 as the departure rate further decreases towards 0, *i.e.* $1/\gamma \rightarrow \infty$. This is intuitive since the decrease in the departure rate means there are more seeds in the system from which the downloaders can download a file. For the given value of λ the downloaders' queue behaves as $M/G/\infty$ for small values of γ when $\gamma < \mu$. Hence, as $1/\gamma \rightarrow \infty$ the mean download time is equal to the mean service time which is 1 for each of the distributions.

The coefficient of variation of the download time as seen in the top right of Fig. 9 stabilizes to a constant value as $1/\gamma$ increases. The values indicate that download

times are similar in distribution to the corresponding service time distribution. The bottom part shows that the mean lifetime of the seeds and the coefficient of variation do not depend upon the distribution of the service time of downloaders which is as expected.

The top part of Fig. 10 presents the mean number of downloaders and its coefficient of variation. The mean value decreases as the departure rate (γ) of the seeds decreases. Decrease in the seed departure rate means the downloaders have more seeds to down-

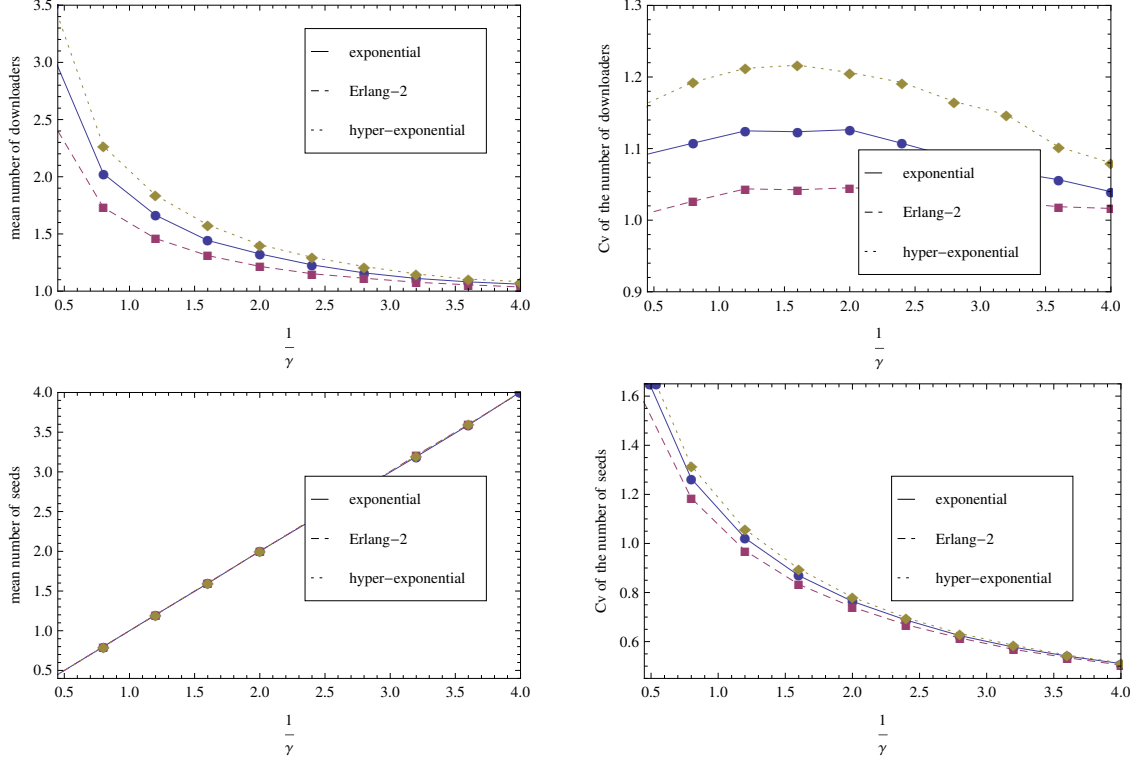


Figure 10: Mean and coefficient of variation of downloaders (C_v) and number of seeds as a function of $\frac{1}{\gamma}$. $\mu = 1$, $\lambda = 1$.

load from which means shorter delays and short queue length. As the departure rate decreases, more seeds accumulate over time which renders the downloaders queue into $M/G/\infty$. Coefficient of variation is seen decreasing towards 1 with the decrease in the seed departure rate. For large values of γ , the distribution depends on the service time distribution of the download time. When the departure rate of seeds becomes very low the mean number of downloaders approaches λ/μ *i.e.* 1 in this case. This shows that the steady state value of the fluid model for the mean number of downloaders is valid if γ is sufficiently smaller than μ .

The bottom part of Fig. 10 shows mean number of seeds and its coefficient of variation. Mean number of seeds increases linearly with $\frac{1}{\gamma}$. Coefficient of variation of the seeds decreases as the seed departure rate decreases. Its value roughly follows $\sqrt{\gamma/\lambda}$ which means the number of downloaders Poisson distributed with parameter λ/γ where λ is 1 in this case.

4.3 Stability Consideration

As discussed in section 2.2, the system parameters λ , μ and γ affect the stability. Fig. 11 shows system metrics such as the mean and coefficient of variation as a function of λ when $\gamma = 2\mu = 2$. In this case where $\gamma > \mu$, the system has stability limits given by $\frac{1}{\lambda} > \frac{1}{\mu} - \frac{1}{\gamma}$. For the given values of γ and μ , $\lambda = 2$ is the stability limit. For the values of $\lambda < 2$, the system is stable as seen in the figure and the values of the metrics are similar to those discussed in section 4.1. However, as the λ approaches 2 the number of downloaders and the mean download time increases rapidly towards large values making the system unstable. When $\lambda > 2$, the downloaders' queue keeps on increasing where as the number of seeds saturates to a constant value corresponding to the departure rate γ .

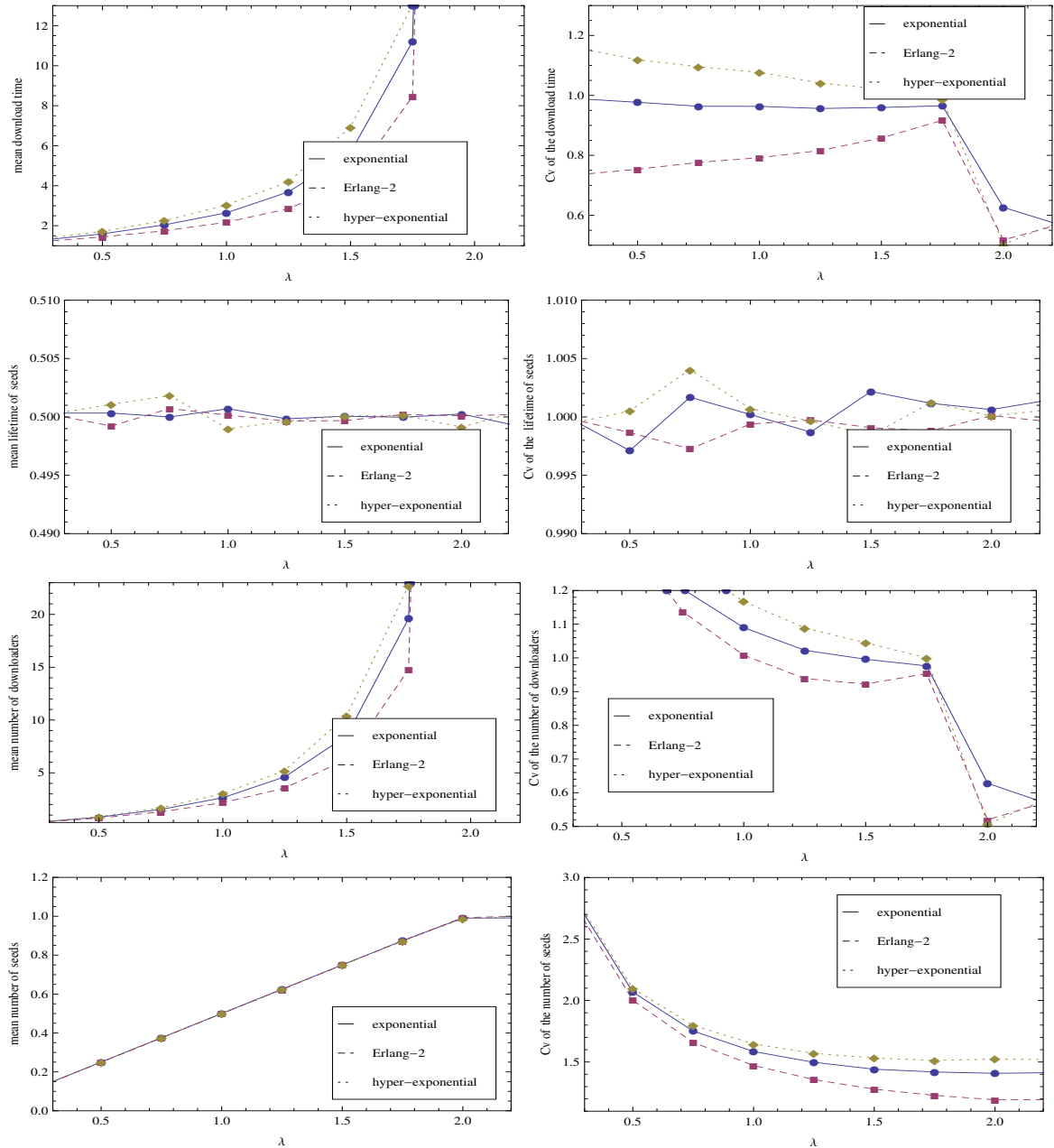


Figure 11: System performance metrics as a function of λ . The departure rate of seeds. $\mu = 1$, $\gamma = 2$.

The value of coefficient of variation of the number of downloaders and the download time takes very small values for all the cases of service time distributions when the system gets unstable. Mean life time of the seeds and its coefficient of variation remains unaffected by the instability of the downloaders' queue as it should be. The mean number of seeds reaches a fixed value depending on the given value of γ and μ and remains constant for increasing values of λ beyond the stability limit. This is so because the number of downloaders changing to seeds is limited by the rate at which the existing seed leave the system when $\gamma > \mu$ no matter what the arrival of downloaders. This means beyond stability limit, the variability in the queue length of remains constant.

The Fig. 12 shows system metrics such as mean and coefficient of variation as a function of $\frac{1}{\gamma}$ when $\lambda = 2\mu = 2$. The system has stability limit given by $\frac{1}{\gamma} > \frac{1}{\mu} - \frac{1}{\lambda}$.

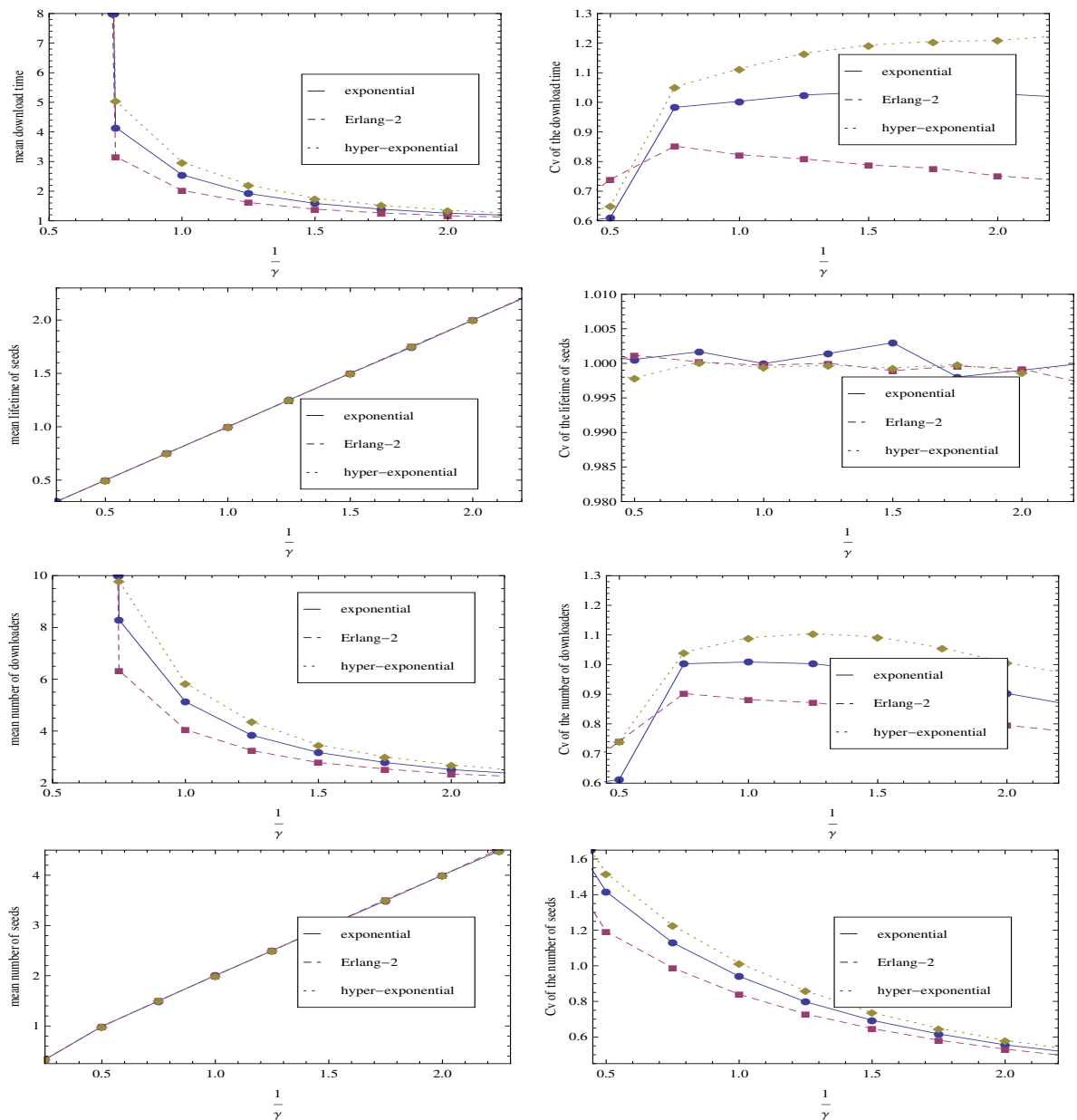


Figure 12: System performance metrics as a function of $\frac{1}{\gamma}$. The departure rate of seeds. $\mu = 1$, $\lambda = 2$.

For the given values of λ and μ , $\frac{1}{\gamma} = 0.5$ is the stability limit. For the values of $\frac{1}{\gamma} < 0.5$ i.e. $\gamma > 2$, the system is unstable as seen in the figure. For the values within the stability limit, the values of the metrics are similar to that as discussed in section 4.2. As $\frac{1}{\gamma}$ approaches 0.5, the number of downloaders and the mean download time increases rapidly towards large values making the system unstable.

The value of coefficient of variation of the number of downloaders and the download time takes very small values for all the cases of service time distributions when the system gets unstable as in the case of varying λ . The mean number of seeds is unaffected by the instability as it only depends upon the mean departure rate for fixed value of λ . The coefficient of variation of seeds follows Poisson distribution for all service time distributions of downloaders. But around the stability limit, the variation depends upon the service time distribution of the downloaders.

4.4 Comparison of Simulation and Numerical Results

Fig. 13 shows the comparison of numerical and simulation results for varying λ in the

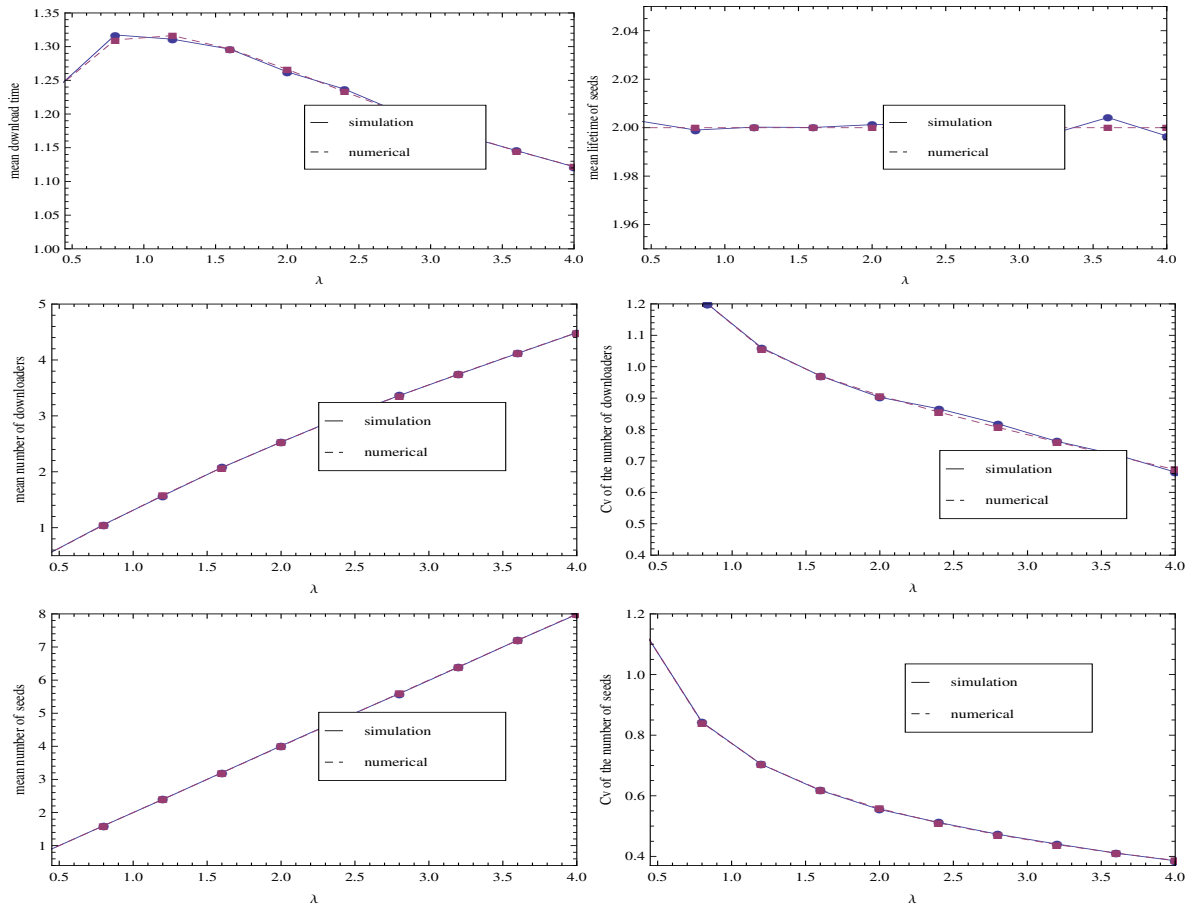


Figure 13: System performance metrics comparing simulation and numerical results as function of λ . for exponential service time distribution. $\mu = 1$, $\gamma = 0.5$.

case $\mu = 1$ and $\gamma = \frac{1}{2}$. The illustrations show the comparison of system metrics for exponential distribution of the service time of the downloaders. The mean download time in the beginning increases with the increase in λ . However with increasing value of λ , it decreases showing that system is scalable. The values of the system metrics

are similar to those discussed in section 4. The figures help to show that the simulation results confirm to the values as predicted by the numerical analysis based on Markovian analysis presented in the section 2.1. The results of the comparison for varying departure rate (γ) is shown below in Fig. 14 for the case $\mu = 1$ and $\lambda = 1$. It also shows the confirmation of simulation results to the numerical analysis using Markovian analysis.

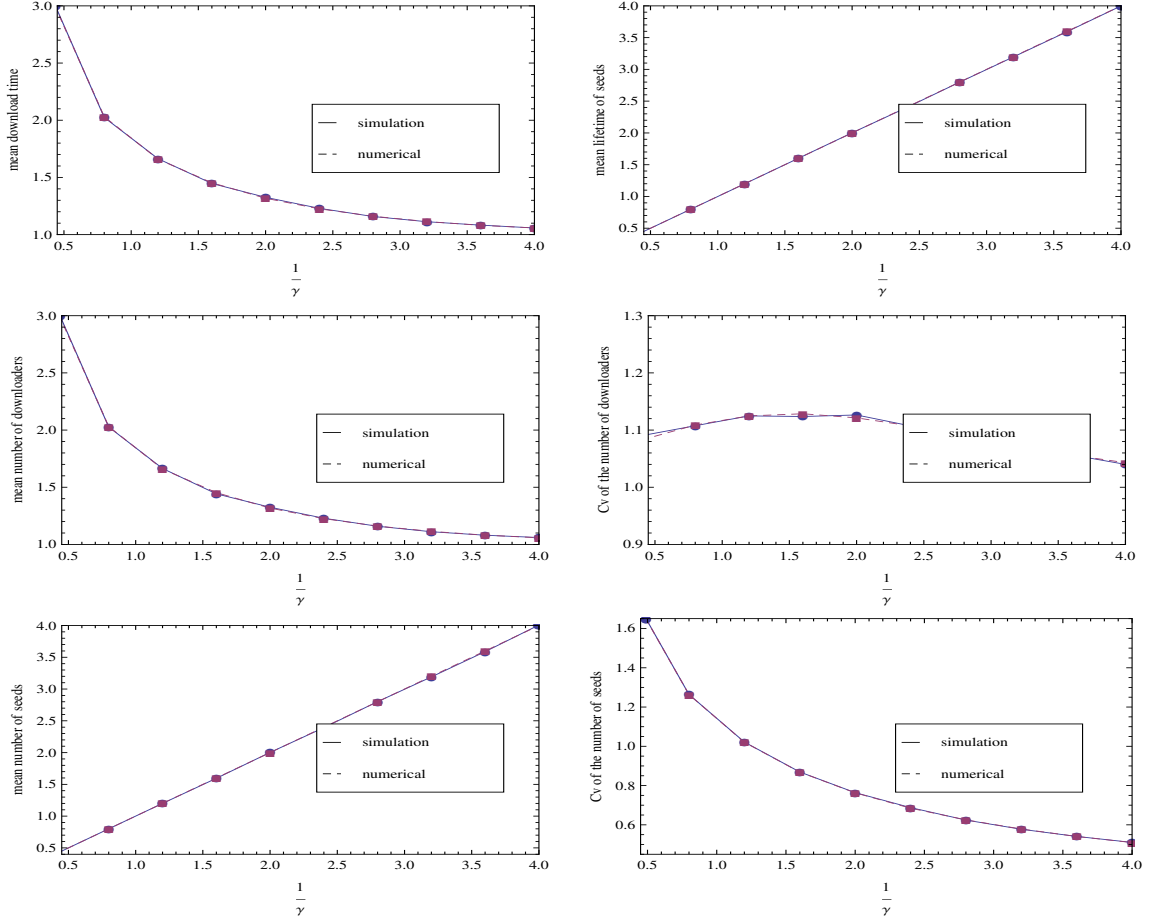


Figure 14: System performance metrics comparing simulation and numerical results as a function of $\frac{1}{\gamma}$. for exponential service time distribution. $\lambda = 1$, $\mu = 1$.

5 Conclusions

In this special assignment we studied the performance parameters of P2P file sharing system using deterministic fluid model and detailed Markov chain analysis. The deterministic fluid model was good in predicting only the steady state values under certain conditions. For example, the steady state value of number of downloaders according to fluid model does not depend on the value γ . However, the Markovian model and its simulation showed that it is so only when the seeds stay in the system sufficiently longer than the service time requested by the downloaders.

The study showed that the service time distribution affects the performance parameters. However, the effect is seen only on the downloading peers. The performance

parameters relating to seeds are seen to be independent of the service time distribution of the downloaders when the system is stable. The study on the stability of the system using simulation showed that the stability limits are same for all the service time distribution. The simulations and the Markovian analysis show that the peer departure rate γ is very crucial to the scalability of the system. For relatively smaller peer departure rate, the performance parameters such as the mean download time are very good. This means strategies have to be used in the design of system which encourage the seeds to stay in the system for long time.

Only the one chunk model was studied in this special assignment. Although this simplifies the analysis of the system, the real applications being deployed are rarely one chunk if any. Nevertheless, the one chunk model helps to understand and observe the properties of P2P systems and it can be extended to the study of multiple chunk file sharing systems in future studies.

References

- [1] 2008 Analysis of traffic demographics in North-American broadband networks, http://www.sandvine.com/general/documents/Traffic_Demographics_NA_Broadband_Networks.pdf
- [2] Internet Study 2007, <http://www.ipoque.com>.
- [3] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, Will IPTV ride the peer-to-peer stream, *IEEE Communication Magazine*, pp. 86-92, June, 2007
- [4] B. Cohen, Incentives build robustness in BitTorrent, in Proc. of First Workshop on Economics of Peer-to-Peer Systems, June 2003.
- [5] K. P. Gummadi, R.J. Dunn, S. Saroiu, S. D. Gribble, H. M. Ley, J. Zahorjan, Measurement, Modeling, and analysis of a peer-to-peer file-sharing workload, in proc. ACM/SPIE conference on multimedia computing and networking, October 2003.
- [6] R. Susitaival, S. Aalto, Modelling the population dynamics and the file availability in a bittorrent-like P2P system with decreasing peer arrival rate, in IWSOS, 2006.
- [7] R. Susitaival, S. Aalto, Analyzing the file availability and download time in a P2P file sharing system, in proceedings of the third EuroNGI conference on the Next Generation Internet Network (NGI), 2007.
- [8] Y. Tian, D.Wu, K.W.Ng, Modeling, analysis and improvement for BitTorrent-like file sharing networks, in INFOCOM 2006.
- [9] D. Qiu, R. Srikant, Modeling and performance analysis of BitTorrent-like peer-to-peer networks, in SIGCOMM 2004.
- [10] L. Kleinrock, *Queueing Systems: Volume I Theory*, New York: Wiley Interscience, 1975.

Appendix A. Markovian Analysis in *Mathematica*:

```
(* m = maximum number of downloaders after truncation *)
(* n = maximum number of seeds after truncation *)
(* eqn = list containing balance equation *)
(* state = list containing steady state probabilities *)
(* solution = list containing steady state probabilities *)
(* λ = arrival rate of downloaders *)
(* μ = service rate *)
(* γ = departure rate of seeds *)

p2psolve[m_, n_, λ_, μ_, γ_] :=
(Module[{sol, solution, min1, min2, p, state = {}, eqn = {}, i, j, sum = 0},
For[i = 1, i < m, i++,
  sum = sum + Subscript[p, i, 0];
  If[n > 0, sum = sum + Subscript[p, i, n];];
  If[n > 0, eqn = Append[eqn,
    ((Subscript[p, i, 0])(λ + μ) - (Subscript[p, i - 1, 0])λ
    - (Subscript[p, i, 1])γ) == 0];];
  min1 = Min[(i + 1), n];
  If[n > 0, eqn = Append[eqn,
    ((Subscript[p, i, n])(λ + n * γ) - (Subscript[p, i - 1, n])λ
    - (Subscript[p, i + 1, n - 1])(min1 * μ)) == 0];];];
For[j = 1, j < n, j++,
  sum = sum + Subscript[p, 0, j];
  If[m > 0, sum = sum + Subscript[p, m, j];];
  If[m > 0, eqn = Append[eqn,
    ((Subscript[p, 0, j])(λ + j * γ) - (Subscript[p, 1, j - 1])μ
    - (Subscript[p, 0, j + 1])(j + 1)γ) == 0];];
  min1 = Min[(j + 1), m];
  If[m > 0, eqn = Append[eqn,
    ((Subscript[p, m, j])(min1 * μ + j * γ) - (Subscript[p, m - 1, j])λ
    - (Subscript[p, m, j + 1])(j + 1)γ) == 0];];];
For[i = 1, i < m, i++, For[j = 1, j < n, j++,
  sum = sum + Subscript[p, i, j];
  min1 = Min[(j + 1), i]; min2 = Min[(i + 1), j];
  eqn = Append[eqn,
    ((Subscript[p, i, j])(λ + min1 * μ + j * γ) - (Subscript[p, i - 1, j])λ
    - (Subscript[p, i + 1, j - 1])(min2 * μ) - (Subscript[p, i, j + 1])(j + 1)γ ==
    0)];];];
sum = sum + Subscript[p, 0, 0];
If[n > 0,
  eqn = Append[eqn, ((Subscript[p, 0, 0])λ - (Subscript[p, 0, 1])γ) == 0];];
If[(m > 0) && (n > 0), eqn = Append[eqn,
  ((Subscript[p, m, 0])(μ) - (Subscript[p, m - 1, 0])λ - (Subscript[p, m, 1])γ) == 0];
  sum = sum + Subscript[p, m, 0];];];
```

```

If[(m > 0)&&(n > 0), eqn = Append[eqn,
    ((Subscript[p, 0, n])(λ + n * γ) - Subscript[p, 1, n - 1]μ) == 0];
    sum = sum + Subscript[p, 0, n];];
If[(n > 0&& m > 0), eqn = Append[eqn,
    ((Subscript[p, m, n])(n * γ) - (Subscript[p, m - 1, n])λ) == 0];
    sum = sum + Subscript[p, m, n];];
eqn = Append[eqn, sum == 1];
For[j = 0, j ≤ n, j++, For[i = 0, i ≤ m, i++,
    state = Append[state, Subscript[p, i, j]];];];
sol = Solve[eqn, state];
solution = Partition[(state/.sol)[[1]], m + 1];
];

```


Appendix B. Simulation of the P2P system in *Mathematica*:

```
(* Exponential random number generator *)
exgenerator[x_]:= $\frac{-\text{Log}[\text{Random}[]]}{x}$ ;

(* Erlang - 2 distributed random number generator *)
erlkgenerator[k_,  $\mu$ _]:= (Module[{i, sum = 0,  $\mu1 = k * \mu$ },
  For[i = 1, i ≤ k, i++, sum = sum + exgenerator[ $\mu1$ ]];
  sum]);

(* Hyper - exponential - 2 random number generator *)
hypegenerator[ $\mu1$ _,  $\mu2$ _, p_]:= (Module[{r, hyp},
  r = Random[];
  If[r < p, hyp = generator[ $\mu1$ ], hyp = exgenerator[ $\mu2$ ]];
  hyp]);

(*  $\lambda$  = arrival rate of downloaders *)
(*  $\mu$  = service rate *)
(*  $\gamma$  = departure rate of seeds *)
simulate[ $\lambda$ _,  $\mu$ _,  $\gamma$ _]:= (Module[{px, py, j, stime, intrx, intry, min,
meanx, (* mean number of the downloaders *)
meany, (* mean number of the seeds *)
stdx, (* mean number of the downloaders *)
stdy, (* mean number of the seeds *)
delayx, (* mean download time *)
delayy, (* mean lifetime of the seeds *)
stddx, (* standard deviation of the download time *)
stddy, (* standard deviation of the lifetime of the seeds *)

(* Initialization *)
i = 0, (* current simulation count *)
imax = 50000, (* number of simulation runs *)
ndx = 0, ndy = 0, (* count of number of departing downloaders and seeds *)
dx = 0, dy = 0, (* cumulative sum of the download time and the lifetime of seeds *)
dxx = 0, dyy = 0, (* cumulative sum of square of the download time and
the lifetime of seeds the lifetime of seeds *)
quex = {}, (* list of downloaders *)
quey = {{0, Infinity}}, (* list of seeds *)
tx = 0, (* arrival or departure time to/from the first queue *),
ty = 0, (* arrival or departure time to/from the second queue * )
x = 0, (* number of customers in 1st Queue *)
y = 0, (* number of customers in 2nd Queue *)
t = 0, (* time *)
intx = 0, intxx = 0, (* cumulative sum of the number of downloaders
and the number of seeds *)
inty = 0, intyy = 0, (* cumulative sum of the square of the number of
```

```

                                downloaders and the number of seeds *)
axtime = exgenerator[λ], (* next arrival time of a downloader *)
dxtime = Infinity,      (* time of next departure of a downloader *)
dytime = Infinity},    (* time of next departure of a seed *)
While[(i < imax),

(* Arrival of a downloader(x) *)
If[((axtime < dxtime)&&(axtime < dytime)),
  t = axtime;
  i = i + 1; quex = Append[quex, {t, exgenerator[μ ]}];
  intx = intx + x * (t - tx);   intxx = intxx + x * x * (t - tx); x++;
  min = Min[x - 1, (1 + y)];
  If[x <= (1 + y), stime = quex[[x, 2]]; px = x; , stime = Infinity; ];
  (* calculate the remaining service time and new departure time of downloader *)
  For[j = 1; , j ≤ min, j++; , quex[[j, 2]] = (tx + quex[[j, 2]] - t) ;
  If[quex[[j, 2]] < stime, stime = quex[[j, 2]]; px = j; ];];
  dxtime = t + stime;
  axtime = t + exgenerator[λ];
  tx = t; Continue[];
];

(* Departure of a downloader(x) and arrival of a seed(y) *)
If[(dxtime < axtime)&&(dxtime < dytime),
  t = dxtime;
  dx = dx + (t - quex[[px, 1]]); dxx = dxx + (t - quex[[px, 1]])2;
  intxx = intxx + x * x * (t - tx); intx = intx + x * (t - tx); x--;
  ndx++; quex = Delete[quex, px];
  intyy = intyy + y * y * (t - ty); inty = inty + y * (t - ty); y++;
  intry = quey[[px]]; quey = Delete[quey, px];
  quey = Insert[quey, intry, Min[1 + x, y]]; (* Update the free server *)
  quey = Append[quey, {t, exgenerator[γ]}];
  (* calculate the remaining service time and new departure time of seed *)
  stime = quey[[ (1 + y), 2]]; py = 1 + y;
  For[j = 1, j ≤ y, j++; , quey[[j, 2]] = (ty + quey[[j, 2]] - t) ;
  If[quey[[j, 2]] < stime, stime = quey[[j, 2]]; py = j; ];];
  dytime = t + stime;
  (* new departure time of the downloaders *)
  min = Min[x, y - 1]; (* Downloaders being served *)
  For[j = 1; , j ≤ min, j++; , quex[[j, 2]] = (tx + quex[[j, 2]] - t) ;];
  min = Min[x, 1 + y];
  For[stime = Infinity; j = 1; , j ≤ min, j++;
  If[quex[[j, 2]] < stime, stime = quex[[j, 2]]; px = j; ];];
  dxtime = t + stime;
  tx = ty = t;
  Continue[];
];

(* Departure of a seed(y) *)

```

```

If[(dytime < axtime)&&(dytime < dxtime),
  t = dytime;
  dy = dy + (t - quey[[py, 1]]); dyy = dyy + (t - quey[[py, 1]])2;
  intyy = intyy + y * y * (t - ty); inty = inty + y * (t - ty); y--; ndy++;
  quey = Delete[quey, py];
(* adjust the position of downloader being served by the departing server *)
min = Min[x, y + 2]; (* New position of the downloader whose seed just left *)
If[py ≤ x, intrx = {quex[[py, 1]], (tx + quex[[py, 2]] - t)};
  quex = Delete[quex, py]; quex = Insert[quex, intrx, min];
  If[x ≤ (1 + y), stime = quex[[x, 2]]; px = x; , stime = Infinity;];
  min = Min[x - 1, 1 + y];
  For[j = 1, j ≤ min, j++, quex[[j, 2]] = (tx + quex[[j, 2]] - t);
    If[quex[[j, 2]] < stime, stime = quex[[j, 2]]; px = j;];];
  intxx = intxx + x * x * (t - tx);
  intx = intx + x * (t - tx); tx = t;
  dxtime = t + stime;];
(* Calculate the remaining service time and new departure time of seeds *)
For[j = 1; stime = Infinity; , j ≤ 1 + y, j++, quey[[j, 2]] = ty + quey[[j, 2]] - t;
  If[quey[[j, 2]] < stime, stime = quey[[j, 2]]; py = j;];];
dytime = t + stime;
ty = t;
];
];
meanx = intx/t;
stddx = Sqrt [(intxx/t) - (meanx)2];
meany = inty/t;
stddy = Sqrt [(intyy/t) - (meany)2];
delayx = dx/ndx;
stdx = Sqrt [(dxx/ndx) - (delayx)2];
delayy = dy/ndy;
stdy = Sqrt [(dyy/ndy) - (delayy)2];
{delayx, stddx, delayy, stddy, meanx, stdx, meany, stdy}
]);
(* End of function simulate *)

```