Modeling Self-organized Application Spreading

Ádám Horváth[1] and Károly Farkas[1,2]

[1] University of West Hungary,
Sopron 9400, Bajcsy-Zsilinszky u. 9., Hungary
{horvath,farkas}@inf.nyme.hu
https://inf.nyme.hu/~{horvath,farkas}
[2] Budapest University of Technology and Economics,
Budapest 1117, Magyar Tudósok krt. 2., Hungary

Abstract. Information spreading in self-organized networks is a frequently investigated research topic today; however, investigating the characteristics of application spreading by exploiting the direct connections between the user devices has not been widely studied yet. In this paper, we present our spreading model, in which we use Closed Queuing Networks to model the application spreading process. In this model, we capture the users' behavior, as well. We also give some simulation results to demonstrate the usage of our model.

Keywords: application spreading, self-organized networks, mathematical modeling.

1 Introduction

Knowing the characteristics of application spreading is important for the application provider, first of all from economic point of view. He has to know or at least assess how much money he can realize from the purchases of a given application in a given time. He should also know, which factors influence the spreading process and how.

Traditionally, applications are distributed via a central entity, like an internet webshop. Users can browse on the internet and select, purchase and download the application software that they like. However, the proliferation of modern communication paradigms, such as self-organized networks can change the characteristics of application spreading. In such networks, users can communicate directly between each other and direct application downloading is available. So, participants of the spontaneous communication can try out applications and get incentives to purchase the ones they liked. The purchasing is available only via a traditional way, since the secure payment in self-organized networks is a challenging issue today.

Application spreading aided by self-organized networks has not got too much attention yet. There are many factors in this type of communication, which are not exploited from economic point of view, such as community experience e.g. with a multi-player game. These factors can have an effect on application

R. Szabó et al. (Eds.): AccessNets 2010, LNICST 63, pp. 71–80, 2011.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2011

spreading, as well, and mean more motivation for users to purchase a given application than he would have seen only some advertisements. Therefore, spontaneous communication can become a new area of the software business.

In this paper, we propose the use of Closed Queuing Networks (CQNs) [1] to model the application spreading aided by self-organized networks. The CQN is a stochastic model, which is appropriate to describe the rapid change of the network topology. We can investigate the application spreading in a given population, which is interested in the spontaneous communication. Moreover, CQNs allow us to order transition intensities to state transitions, by which we can describe the time behavior of the spreading process.

The technical details of the application spreading, such as discovering nodes, deploying, managing and terminating the application software are beyond the scope of this paper. These issues are detailed in other contributions, e.g. in [2]. Similarly, we do not focus on security issues, which can be found in other works, e.g. in [3], [4], [5] and [6].

The rest of the paper is organized as follows. In Section 2, we present our communication model. We describe our spreading model based on CQNs in Section 3. We present some simulation results in Section 4. We give an overview about the related works in Section 5. Finally, we give a short summary in Section 6.

2 Communication Model

To model the spreading process, we use the following communication model. We examine the spreading of a given application, which is a multi-user application having two versions, a trial and a full version. We examine the application spreading in a given population, which is composed of the users, which are interested in the use of the application. The uninterested individuals do not influence the spreading process, so we do not consider them as users. Henceforth, we refer to the investigated population simply as users.

We can categorize the users into different classes depending on whether they possess any versions of the application or not. Since our model shows similarities with epidemic spreading models, we named the different classes after the terminology of epidemics. We call a user infected, if he possesses the full version of the application, susceptible, if he possesses the trial version of it and resistant, if he possesses none of them, or he has already lost his interest of using it.

Users communicate with each other forming self-organized networks from time to time, and direct communication takes places between them. The users can download the trial version of the application and try it out only if there is at least one infected user in the same network. If a user liked the application, he can purchase it using a traditional purchasing way, e.g., using a webshop on the Internet (this phase is necessary, since the secure payment and licensing method in self-organized environment is a challenging issue). Later, if a user purchased the application, he can use it or even spread its trial version further.

Susceptible users will be motivated in purchasing the full version of the application only if there are limitations in using the trial version. Hence, we apply a

limit (leech limit) that restricts how many susceptible nodes (leech[1]) can connect to one infected node (seed[1]). In this sense, we can consider the seeds as servers, which can serve a limited number of clients. A seed must be an infected user, while a leech may be either infected or susceptible. Fig.1 shows the case when a susceptible user purchases the application. The light and the dark laptops depict susceptible users, while the PDA depicts an infected user. In this example, the leech limit is two, so only two susceptible users (the light laptops) can connect to the only infected user and two users (the dark laptops) have to wait. After one of them purchased the application, they can use it, too.

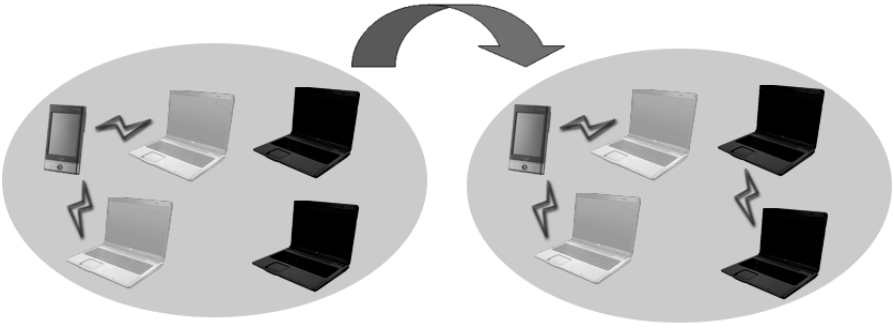

Fig. 1. Change of application usage when a susceptible user purchases the application

Furthermore, we distinguish three different user types based on their behavior. $Type_A$ users are interested in using the application; so if they liked it they will purchase it, possibly without trying it out. $Type_B$ users are also interested in using the application, but they will purchase the application with a given intensity only if they cannot find a seed from time to time which they can connect to. However, $Type_C$ users never buy the application, their presence influences the spreading process.

3 Modeling Application Spreading

In this section, we present our spreading model and describe how we can use it.

3.1 Spreading Model

We use CQNs for modeling the spreading mechanism, because it is appropriate for describing stochastic processes. Moreover, we can define the transition intensities of the state transitions, what allows us to investigate the time behavior of the spreading process.

[1] After the terminology of BitTorrent [7].

In our CQN model, the different states represent the whole user population (Fig. 2). Each user is in one state, and the users' state depends on their actual user class.

The resistant users, which possess neither the trial version of the application nor the full version of it are in state 0. We call also resistant the users, which possess either the trial version (state 5) or the full version (state 6) of the application, but already lost the interest in using it. The susceptible users, which are currently not using the application (passive susceptibles) are in state 1, while susceptible users, which are currently using the application (active susceptibles) are in state 2. Similarly, the passive and active infected users are in state 3 and 4, respectively.

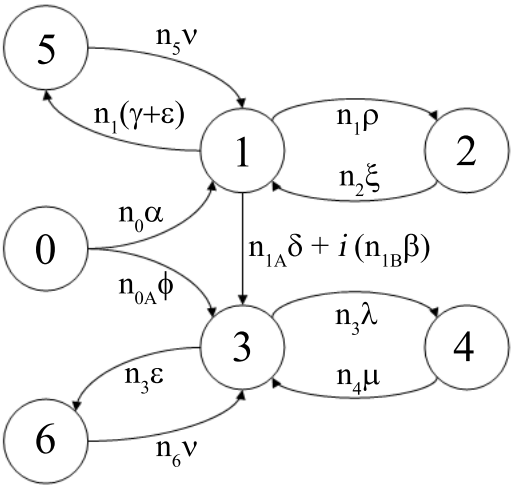

Fig. 2. The proposed CQN model for application spreading

The Greek letters in Fig. 2 represent the transition intensities of a single user, n_x represents the number of users in state x, while n_{xA}, n_{xB} and n_{xC} represent the number of $Type_A$, $Type_B$ and $Type_C$ users in state x, respectively ($n_x = n_{xA} + n_{xB} + n_{xC}$). The transition intensities are real numbers assigned to the state transitions, and denote how many times a state transition takes place during a given time interval on average. The transitions of our model are described in Table 1.

In our model, we do not consider the network topology as a key element of the application spreading, we assume that users can connect to each other forming self-organized networks from time to time. They change their state depending on whether they have got any version of the application, or whether they start to run the application or stop using it.

Table 1. Description of the State Transitions in our CQN

Transitions	Description
$1 \rightarrow 2$	A susceptible user starts to run his application and tries to connect to an available seed in the network. If he cannot find one, he has to wait.
$2 \rightarrow 1$	A susceptible user stops running the application.
$3 \rightarrow 4$	An infected user starts to run the application. he can either try to connect to an available seed, or will be a seed himself to which leeches can connect.
$4 \rightarrow 3$	An infected user stops running the application.
$0 \rightarrow 1$	A resistant user downloads the trial version of the application.
$3 \rightarrow 6$	An infected user loses the interest in using the application.
$1 \rightarrow 5$	A susceptible user loses the interest in using the application. The additional intensity γ represents that susceptible users lose the interest faster because they have to wait possibly.
$0 \rightarrow 3$	A resistant user purchases the application. It is possible that someone purchases it without trying it out. In our model, we allow this state transition only to $Type_A$ users, thus, n_{0A} depicts the number of $Type_A$ users in state 0.
$1 \rightarrow 3$	A susceptible user purchases the application. n_{1A} and n_{1B} depict the number of $Type_A$ and $Type_B$ users in state 1, respectively. This transition is enabled for $Type_B$ users only when they cannot find a free seed in the network to which they can connect. Therefore, the indicator variable i is one if at least one free seed is available, and zero otherwise. Since $Type_C$ users never purchase the application, this transition is not allowed to take place for them. However, $Type_C$ users can also connect to seeds, so their presence decreases the probability that $Type_B$ users can find a free seed.
$5 \rightarrow 1$	A resistant user, which lost the interest in using the trial version, wants to use the application again after a while. His state changes to susceptible.
$6 \rightarrow 3$	Similarly, if a resistant user, which possesses the full version of the application, so his state changes to infected.

3.2 Usage of the Spreading Model

We can unambiguously describe the system by the user distribution (n_0, n_1, n_2, n_3, n_4, n_5, n_6). The transition intensity values with regard to a single user (α, β, γ, δ, ϵ, ϕ, λ, μ, ν, ρ and ξ) are the parameters of our model, which we can set experimentally. The holding time h can be computed in each system state by the following way:

$$h = \frac{1}{\sum\limits_{\forall state\ x} out_x}, \tag{1}$$

where out_x depicts the sum of intensities for transitions leaving state x. The holding time h is the time that the system spends in a given system state. We compute it in each system state, since the transition intensities, from which we derive it, change after each transition due to the user distribution changes. The next system state can be generated based on the ratio of the transition intensity values. For example, the probability that the next transition in the system will be transition $1 \rightarrow 2$ is $n_1 \cdot \rho \cdot h$. After we selected the transition that takes place, we move one user from the source state to the destination state of that transition and compute the holding time of the new system state, and so on.

If a user reaches state 5 or 6, it means that he lost the interest in using the application. We allow users to return also from these states, since it is possible, that they will be interested later again. However, we allow these transitions with low intensity values to ensure that the system will reach sooner or later the state (final system state) in which all users are in either of state 5 and state 6. However, the final system state is not the steady state of the system, our investigations will finish when we reach it, because all users lost the interest in using the application. In this state, the holding time is very large, since we set ν to low. Therefore, the further transitions need very much time to take place, so the interest in using and purchasing the application is very low.

The number of transition $1 \rightarrow 3$ and $0 \rightarrow 3$ will determine how many pieces of the application software were sold until we reached the final system state, while the duration of the spreading process can be determined by summing the holding times. By using a simulator software, we can evaluate the simulation results and learn the dynamics of the spreading process and the attitude of the different user types.

4 Simulation Results

In this section, we present some simulation results to demonstrate how the spreading mechanism takes place.

To get the simulation results, we have developed a Java based simulator software. The simulator works as follows. In the initial system state, when every node is in state 0, it computes the holding time and stores it. The next system state is generated by selecting the transition that takes place. The selection works by using random numbers, which are weighted with the intensity values of the transitions. The simulator moves one user from the source state to the destination state of the selected transition. We compute and store the holding in the next system state, as well, and so on. The whole process terminates when we reach the final system state first. In every system state, we can store statistics and combining them with the holding time we can investigate the time behavior of the spreading process.

In the following, we describe two scenarios, in which we ran the simulator with different parameters. The parameters are collected in Table 2. The measure of the parameters depicted by Greek letters in Table 2 is 1 / hour.

Table 2. Simulation Parameters

Parameters	Simulation 1	Simulation 2
α	10^{-3}	$3 \cdot 10^{-3}$
β	10^{-3}	10^{-3}
γ	10^{-3}	10^{-3}
δ	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
ϵ	10^{-3}	10^{-3}
ϕ	10^{-5}	10^{-5}
λ	$2.5 \cdot 10^{-2}$	$2.5 \cdot 10^{-2}$
μ	1	1
ρ	$2.5 \cdot 10^{-2}$	$2.5 \cdot 10^{-2}$
ξ	1	1
ν	10^{-9}	10^{-9}
n_A	300	300
n_B	400	400
n_C	300	0
leech limit	2	2

In this paper, we do not aim to find the correct set of the parameters, which is a complex and hard challenge, and we plan to do it in the future. The parameters depend on many things, such as the popularity or the price of the given application, thus, they can be set experimentally. In these simulations, we tried to set the parameters as realistic as possible based on good sense. For example, $\lambda = 2.5 \cdot 10^{-2}/hour$ means that an infected user starts the application once in every 40 hours on average, while $\mu = 1/hour$ means that he is using it for 1 hour on average. In both simulation runs, we repeated the simulations 10 times, and we got similar results. Therefore, we randomly picked up one in both cases for investigations.

In Simulation 1, we got that 3 users purchased the application without trying it out, 215 $Type_A$ user and 74 $Type_B$ user purchased it after trying it out. Thus, the total number of purchases was 292. The duration of the process was about 8000 hours, which is almost one year; however, the interest to purchase the application was very low after 5300 hours (Fig. 3).

In Simulation 2, we set the number of $Type_C$ users to zero to demonstrate how they influence the number of purchases of $Type_B$ users. Moreover, we tripled the value of transition intensity α, which will fasten the spreading of the trial version, and the whole spreading process. Fig. 4 shows that the number of purchases of $Type_B$ users decreased to 63, therefore, the total number of purchases decreased to 280. It can be explained by the absence of $Type_C$ users: $Type_B$ users found more frequently an available seed to connect to, and their motivation to purchase the application decreased. Since we increased the value of transition α, the simulation reached the final system state after 6270 hours.

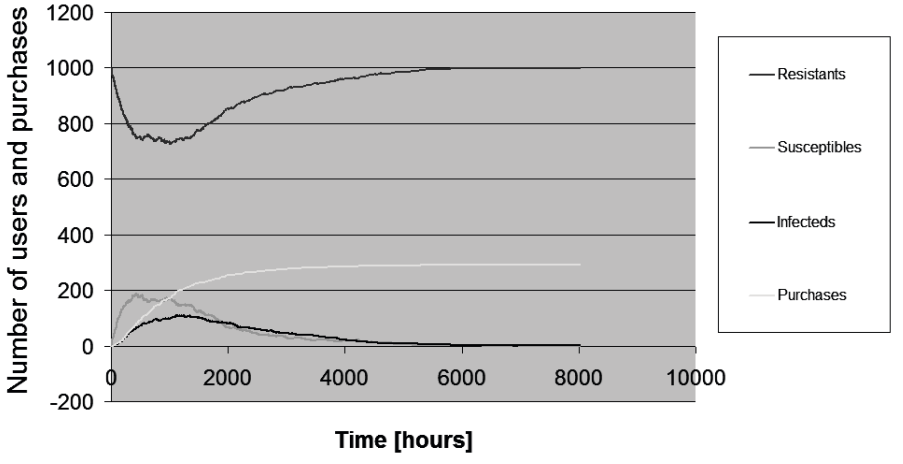

Fig. 3. The size of user classes and the number of purchases – Simulation 1

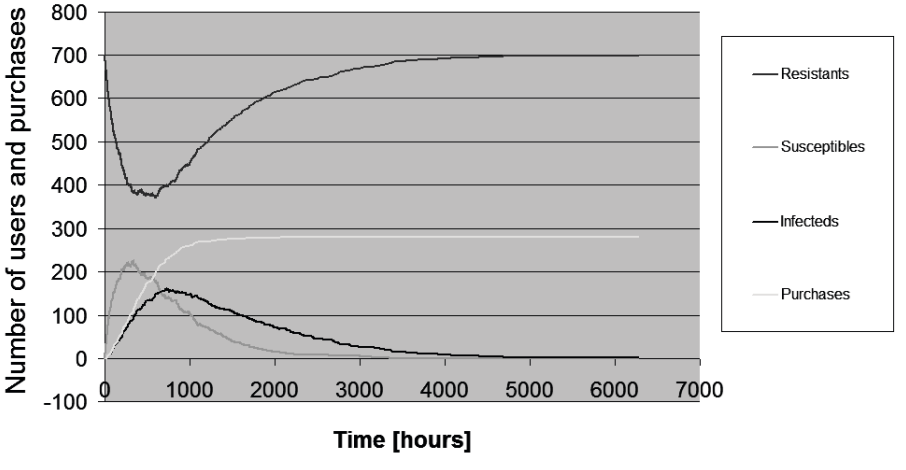

Fig. 4. The size of user classes and the number of purchases – Simulation 2

5 Related Works

Investigating application spreading has not got too much attention so far; however, it becomes more and more important with the proliferation of the modern communication paradigms. On the other hand, epidemic spreading is a popular research topic today and deals with issues similar to ours.

In [8], the authors investigate the propagation of a virus in a real network. They present a model to determine an epidemic threshold in the network, below which the number of infected nodes decreases exponentially. The threshold is derived from the adjacency matrix of the network. In [9], the authors use scale-free networks to model the spreading of computer viruses and also give

an epidemic threshold, which is an infection rate. Information spreading is also modeled by epidemic spreading models, such as the susceptible-infected-resistant (SIR) model [10], or other models based on the network topology [11], [12]. In [13], the spreading of malicious software over mobile ad hoc networks is investigated. The authors propose the use of the susceptible-infected-susceptible (SIS) model on the basis of the theory of closed queuing networks.

In [14], the authors propose the commercial use of ad hoc networks and present a radio dispatch system using mobile ad hoc communication. In the proposed system, the network topology is the key element of the information dissemination.

In our model, we do not consider the network topology as a key element of the application spreading, since no real-time information dissemination is needed between the users. The above mentioned papers do not deal with the application spreading, do not capture the users' behavior, and except [14], they do not touch the commercial use of the self-organized networks, in which the authors consider the information dissemination as a tool, not as a goal.

6 Summary

In this paper, we investigated the application spreading aided by spontaneous communication. We proposed a CQN model to describe the application spreading process, in which we assumed that users support the spreading process by the distribution of the trial version of an application. We categorized the users into different classes based on their behavior. Finally, we gave some simulation results to demonstrate the usage of our model.

In the future, we plan to elaborate on setting the model parameters as realistic as possible and investigate other tools, such as Stochastic Petri Nets, to be able to refine our spreading model.

Acknowledgements

This work has been partially supported by the Hungarian Scientific Research Fund (OTKA, PD 72984).

References

1. Robertazzi, T.G.: Computer Networks and Systems: Queuing Theory and Performance Evaluation. Springer, New York (1994)
2. Plattner, B., Farkas, K.: Supporting Real-Time Applications in Mobile Mesh Networks. In: MeshNets 2005 Workshop, Budapest, Hungary (2005)
3. Čapkun, S., Buttyán, L., Hubaux, J.-P.: Self-Organized Public-Key Manegement for Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing 2(1) (2006)
4. Hu, Y.-C., Johnson, D.B., Perrig, A.: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In: 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), Callicoon, New York, USA (2002)

5. Hu, Y.-C., Perrig, A., Johnson, D.B.: Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In: 8th ACM International Conference on Mobile Computing and Networking (MobiCom), Atlanta, Georgia, USA (2002)
6. Sanzgiri, K., Dahill, B., Levine, B.N., Shields, C., Belding-Royer, E.M.: A Secure Routing Protocol for Ad Hoc Networks. In: 10th IEEE International Conference on Network Protocols (ICNP), Paris, France (2002)
7. Cohen, B.: Incentives Build Robustness in BitTorrent. In: 1st Workshop on Economics of Peer-to-Peer Systems, UC Berkeley, California, USA (2003)
8. Wang, Y., Chakrabarti, D., Wang, C., Faloutsos, C.: Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint. In: 22nd International Symposium on Reliable Distributed Systems (SRDS 2003), Florence, Italy, pp. 25–34 (2003)
9. Pastor-Satorras, R., Vespignani, A.: Epidemic Spreading in Scale-Free Networks. Phys. Rev. Lett. 86, 3200 (2001)
10. Fu, F., Liu, L., Wang, L.: Information Propagation in a Novel Hierarchical Network. In: 46th IEEE Conference on Decision and Control, New Orleans, USA (2007)
11. Khelil, A., Becker, C., Tian, J., Rothermel, K.: An Epidemic Model for Information Diffusion in MANETs. In: 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, Atlanta, Georgia, USA (2002)
12. Sekkas, O., Piguet, D., Anagnostopoulos, C., Kotsakos, D., Alyfantis, D., Kassapoglou-Faist, C., Hadjiethymiades, S.: Probabilistic Information Dissemination for MANETs: the IPAC Approach. In: 20th Tyrrhenian Workshop on Digital Communications, Pula, Italy (2009)
13. Karyotis, V., Kakalis, A., Papavassiliou, S.: Malware-Propagative Mobile Ad Hoc Networks:Asymptotic Behavior Analysis. Journal of Computer Science and Technology 23(3), 389–399 (2008)
14. Huang, E., Hu, W., Crowcroft, J., Wassel, I.: Towards Commercial Mobile Ad Hoc Network Application: A Radio Dispatch System. In: 9th Annual International Conference on Mobile Computing and Networking, San Diego, California, USA (2003)