**Bond University**
**ePublications@bond**

Information Technology papers | Faculty of Business

11-6-2000

# Design and Analysis of Code Distribution System With Active Networks

Zheng da Wu
*Bond University*, Zheng_Da_Wu@bond.edu.au

X. G. Wang
*Institute of Computing Technology, Beijing, China*, ccss@ict.ac.cn

Follow this and additional works at: http://epublications.bond.edu.au/infotech_pubs

Recommended Citation
Zheng da Wu and X. G. Wang. (2000) "Design and Analysis of Code Distribution System With Active Networks".Nov. 2000.

http://epublications.bond.edu.au/infotech_pubs/25

This Conference Paper is brought to you by the Faculty of Business at ePublications@bond. It has been accepted for inclusion in Information Technology papers by an authorized administrator of ePublications@bond. For more information, please contact Bond University's Repository Coordinator.

# Design and Analysis of Code Distribution Systems With Active Networks

Z. D. Wu and X.G. Wang[+]
School of Information Technology, Bond University,
Gold Coast, Qld 4229, Australia. E-mail: wz@bond.edu.au
[+] Institute of Computing Technology, The Chinese Academy of Sciences,
Beijing, China. E-mail: ccss@ict.ac.cn

**Abstract**

In this paper an analytical queueing model is proposed based on our study of the code distribution and especially in the integrated or in-band code distribution systems with active networks. By using this model the system performance and reliability can be evaluated so as to provide insights into their behaviour. Since there are different proposals for the code distribution reported recently, it is significant to provide a technique to find an optimal decision in the network design. A procedure to help for this decision is presented. Some numerical results of using our performance model and decision procedure are also demonstrated in the paper.

**Keywords:** *Active Networks, Code Distribution, System Performance and Reliability, Queueing Models.*

## 1. Introduction

As the Internet explodes on size, service type, and the number of users, it has been identified several problems with today's Internet : the difficulty of integrating new technologies and standards into shared network infrastructure, poor performance due to redundant operations at several protocol layers, and difficulty accommodating new services, such as multimedia applications, in the existing architectural model. It is noted that the RSVP, Mobile IP, and IPv6 have experienced long standardization process and it is still difficult to make them in common use [1]. In order to solve these problems, a concept of *active networking* was introduced by the DARPA research community in 1994-1995. This concept breaks with traditional networking model by allowing the network to perform customized computations on the user data of packets when they travel through intermediate nodes, sometimes called "store - compute and forward" mode. As a guideline, the DARPA has defined architectural framework for active networks. Briefly, it consists of four layers, transmission facilities, node operating system, execution environment (EE), and applications. Many research in active network architectures have been reported recently, such as ANTS, SwitchWare, M0, Smart-Packets, ASP, Liquid Software, CANES, Protocol Boosters and NetScript [1]-[6]. It has been recognized that there are two big challenges in the development of active networks: security and performance. This paper deals with some performance issues of active networks. Since code distribution is a new kind of communication mechanisms that must be employed in active networks and directly affects the network performance, it is significant to investigate its behaviour during an active network design. The description of these systems are given in Section 2. In Section 3 we propose an analytical queueing model based on our classification of code distribution approaches, called push-based and pull-based system. By using this model the system performance and reliability can be evaluated so as to provide deep insights into their behaviour. Since there are different proposals for code distribution reported in the literature, it is necessary to provide a technique to find an optimal decision in the network design. As a result, a procedure to help for this decision is developed in Section 4. We demonstrate some numerical results of using our performance model and decision procedure in Section 5.

## 2. System Description

In order to customize network behaviour and make a network "active", user programs or codes are required to be distributed to a set of active nodes where they are executed. There are two approaches for code distribution as classified in [1]:

1. Discrete or out-of-band approach

In this approach, users inject custom programs into the nodes through a separate, auxiliary, out-of-band mechanism. When a packet arrives at a node, the corresponding program is selected based on setting in the packet header and the packet's payload is processed accordingly. Programming of switches or nodes can happen

in advance of packet transmission or on-demand upon packet arrival [6]. This approach is preferable when the programs are large compared to packet size. It also maintain modularization between user data and program, enabling better control by network administrator over what programs are loaded into the nodes.

2. Integrated or in-band approach

This scheme views every packet as a program. Individual packets, called *capsules* [2] or *active packets* [3], carry the protocol, in the form of code, and the associated data with them as they travel through the network. Upon arrival at a node, the packet contents are dispatched to selected execution environment (EE) where they can be executed safely. This approach does not require any storage and have a limited time-to-live (TTL) at nodes. ANTS and Smart-Packet are examples of using this method for code distribution [2][5].

Of course, a hybrid of the two approaches can be employed in an active network design. A typical example is with SwitchWare [3]. Since the first approach is quite similar to existing code loading systems for installation or update of protocols at routers or gateways. It is not our emphasis in this paper. Alternatively, the second approach is a quite new communication mechanism that plays one of key roles in active networks. Thus it is necessary to investigate its performance behavior and reliability. In order to give the system description more explicit for the integrated or in-band approach, we further classify such kind of systems into two types, simply called *push-based* system and *pull-based* system. In the case of push-based system, code is injected or pushed into a set of nodes by a given node in random or periodical way. The given node is regarded as code *supplier* in this paper. In other words, with a push-based system codes are actively pushed by the supplier. It can be illustrated as in Figure 1 where a supplier pushes codes into a set of nodes by means of sending active packets, which carry the codes, to them , and then the codes are executed in these nodes.

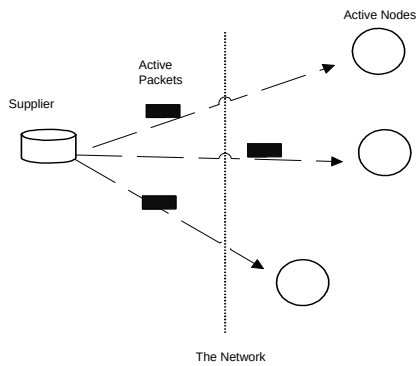


Figure 1: Example for Push-Based System

In a pull-based system, code is pulled or obtained from somewhere passively. This occurs when a node receives an active packet, but the node cannot perform computation on the packet due to the related code not facilitated in the node yet. In other words, code is on-demand upon a packet arrival in such system. An example of the pull-based distribution is shown as in Figure 2, where a packet arrives at node $i$ from node $i-1$, but node $i$ cannot execute it and has to get its related code from somewhere. This can be achieved from either "previous node", node $i-1$, or well-known supplier.

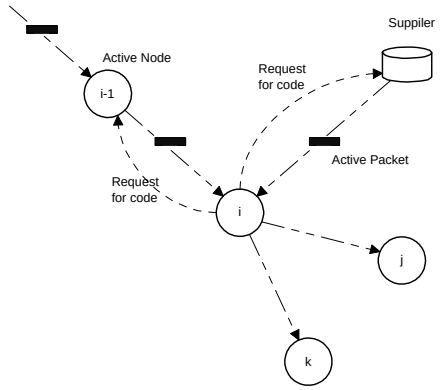


Figure 2: Example for Pull-Based System

## 3. Analytical Model

In this section we propose an analytical model for code-distribution systems, which can be applied to the both push and pull-based distribution systems. The model provides a base on which the system performance and reliability may be evaluated.

Assuming that each application within an active network is constructed by a set of active routines (programs or codes) and they will be injected into one or more nodes by using a code distribution system. These active routines are organized with their names or identifiers [2][3]. Thus, each active packet carries a pair of items {ID, Routine}, where ID is a reference to the routine. Code supplier $S$ sends active packets with either push or pull-based distribution approach. We denote $A(t)$ as the set of all active routines for application $A$ at any given time $t$, which are stored in supplier $S$. During the transmission of the packets, some of them might be corrupted or lost, which have to be retransmitted. When an active routine is accepted by an active node without any error in transmission, the routine is executed with two possibilities: success or failure. This depends on other conditions in the node, such as the status of its execution environment (EE) and resource availability.

We are interested in two performance measures for code-distribution systems (i) system consistency; (ii) system latency. If the set of active routines issued by a

supplier is successfully accepted by its destination node, this code deployment is thought as *consistent*, otherwise, called *inconsistent*. There are two inconsistent cases: one is caused by packets-loss during their transmission and another is that the destination node is in "starving' state. It means that when an active packet arrives at the node it cannot be executed due to it's code or execution environment not facilitated in the node at that time. In other words, the node is demanding for new or update routines from some supplier so as to continue the application. Usually, this occurs when the pull-based distribution approach is used for some network applications [2]. Assuming that active packets are sent from supplier $S$ to active node $D$ over a loss-prone network channel. Each active packet is associated with a pair $\{i, r(i)\}$, where $i$ is an identifier to routine $r(i)$. The consistency, $c(i,t)$, of packet $\{i, r(i)\}$ is defined as the probability that both $S$ and $D$ have the exactly same routine or code for a given identifier at instant $t$. This can be expressed as

$$c(i,t) = P[S(r(i)) = D(r(i))], \quad 0 \leq c(i,t) \leq 1$$

where $S(r(i))$ and $D(r(i))$ represent routines facilitated in both node $S$ and $D$ respectively at time $t$. Recall the definition of $A(t)$ above, the instantaneous system consistency, $c(t)$, can be defined as the average consistency of all active routines with application $A$ at instant $t$ as below

$$c(t) = \frac{\sum_{i \epsilon A(t)} c(i,t)}{|A(t)|}$$

Consequently, the average system consistency, $E[c(t)]$, is the time average of instantaneous system consistency over the entire life of the system and it is defined as

$$E[c(t)] = lim_{T \to \infty} \frac{\int_{t=0}^{t=T} c(t)dt}{T}$$

It notes that $E[c(t)]$ is one of important metrics to evaluate code-distribution systems. Another interesting measure, the average system latency, $\bar{R}$ (simply called the latency or system latency in this paper), is defined as the average time from the instant a new or update active packet $\{i, r(i)\}$ is injected into the system from its supplier until the first time it is accepted successfully at its destination node.

In order to evaluate the two performance measures defined as above, we propose an analytical model based on "Open, Closed, and Multiple Queueing Network with Multiple Class Jobs" [7][8] and the routing probabilities of traffic in the queueing system are shown as in Figure 3.

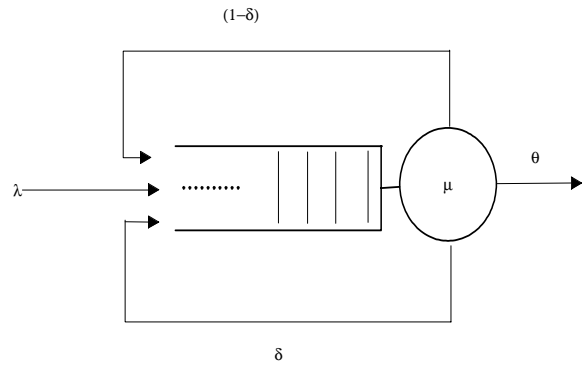


Figure 3: Queueing Model

There are three classes of traffic in the system described as below: (i) external workload with average arrival rate $\lambda$ represents the active packets flow which carry original or new routines from a supplier to its destination node; (ii) during the transmission of the packets, some of them might be lost or corrupted with loss rate $\delta$, $0 \leq \delta \leq 1$, and the error packets will be retransmitted as one feedback traffic in the system; (iii) Another feedback traffic with probability $(1 - \delta)$ represents the packets which update their input data or simply repeat calls for their routines without any change of the original codes. In fact these routines have already run successfully in the destination node. In other words, such active packets only supply code ID with or without new input data.

Recall the concept of system consistency defined as before, it notes that traffic (i) and (ii) deal with inconsistent traffic, denoted as $\lambda_I$ , and traffic (iii) is referred to consistent traffic, as $\lambda_C$. According to the research outcomes of active networks to date, the most of active routines have a limited or even very short lifetime within a node where they are executed [2][3]. Thus, the routines eventually exit from the system with death probability $\theta$. The service rate $\mu$ depends on a number of characteristics of the system, such as the available channel bandwidth, denoted as $b$, the length of packets, $l$, and the set-up time for the context of code execution at the node, $s$. Thus, $\mu$ can be expressed as a function of them, $\mu = f(b, l, s)$.

In order to evaluate the system performance and reliability using this queueing model, we consider the system with two states, consistent and inconsistent, denoted as $C$ and $I$. The probabilities of state change between them is described in the table as below:

| | $I(Exit)$ | $C(Exit)$ | $Death$ |
|---|---|---|---|
| $C(Enter)$ | 0 | $(1-\theta)$ | $\theta$ |
| $I(Enter)$ | $\delta(1-\theta)$ | $(1-\theta)(1-\delta)$ | $\theta$ |

Thus, we can compute the two flows, $\lambda_I$ and $\lambda_C$, in the system by

$$\lambda_I = \lambda + \lambda_I\ (1-\theta)\delta \text{ and}$$
$$\lambda_C = \lambda_C(1-\theta) + \lambda_I\ (1-\delta)(1-\theta).$$

Solving the two equations above yields

$$\lambda_I = \frac{\lambda}{1-\delta(1-\theta)} \quad (1)$$
$$\lambda_C = \frac{\lambda(1-\theta)(1-\delta)}{\theta(1-\delta(1-\theta))} \quad (2)$$

Based on the Jackson's result [7][8], the joint probability distributions of the number of consistent jobs (here packets), $n_C$, and inconsistent jobs, $n_I$, can be expressed as

$$p(n_C, n_I) = \frac{(n_C+n_I)!}{n_C!n_I!}\frac{\lambda_C^{n_C}\lambda_I^{n_I}}{(\lambda_C+\lambda_I\ )^{(n_C+n_I)}}(1-\rho)^{(n_C+n_I)} \quad (3)$$

where $\rho = \frac{\lambda_I+\lambda_C}{\mu}$. The solution is valid only if $\rho < 1$. As a result, the average system consistency, $E[c(t)]$, is given by

$$\begin{aligned} E(c(t)] &= \sum_{n_C+n_I>0} p(n_C, n_I)\frac{n_C}{n_C+n_I} \\ &= \sum_{n_C+n_I=k\leq 0} \frac{\lambda_C}{\lambda_I\ +\lambda_C}(1-\rho)\rho^{k+1} \\ &= \frac{\lambda_C}{\mu} \\ &= \frac{\lambda(1-\theta)(1-\delta)}{\theta(1-\delta(1-\theta))\mu} \end{aligned} \quad (4)$$

Therefore, once $\lambda$, $\theta$, $\delta$ and $\mu$ are specified we can compute the average system consistency $E[c(t)]$ directly from equation (4).

We employ the Mean Value Analysis (MVA) solution technique to evaluate the average system latency, $\bar{R}$, for open queueing networks with multiple job classes [9]. According to this technique, the average latency of jobs with class $i$ at node $k$ can be determined by

$$R_{i,k}(\vec{\lambda}) = \frac{D_{i,k}}{1-\sum_{j=1}^{M} U_{j,k}(\vec{\lambda})} \quad (5)$$

and

$$U_{j,k}(\vec{\lambda}) = \lambda_j D_{j,k} \quad (6)$$

where $M$ is the total number of job classes; $\vec{\lambda} = (\lambda_1, \lambda_2, ..., \lambda_M)$ is the vector of arrival rates with all classes and $D_{i,k}$ is the service demand or time of job class $i$ at node $k$. This solution is restricted to $max\{\sum_{j=1}^{M}\lambda_j D_{j,k}\} < 1$ for each node $k$. Thus, we can apply formula (5) and (6) to compute $\bar{R}$ now. As our model is a special open queueing network with only one service center and three classes of jobs, we only need to specify two vectors $\vec{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ and $\vec{D} = (D_1, D_2, D_3,)$. This can be achieved by using the following assignments: $\lambda_1 = \lambda$, $\lambda_2 = \lambda_I - \lambda$, and $\lambda_3 = \lambda_C$. Usually, $\vec{D}$ can be specified by users directly according to the system characteristics. Based on equations (1) and (2) given as above, $\bar{R}$ can be evaluated easily. We will demonstrate some numerical results of using this mode for the performance analysis with its reliability in Section 5.

## 4. Optimal Decision

There are two schemes for in-band code distribution with active networks, push-based and pull-based, as discussed in Section 2. These lead to two questions. The first question is how to find out the best location of the code supplier which will benefit all users (or nodes) in the network if the push-based scheme is chosen. Alternatively, if the pull-based scheme is selected, there are two specific methods for code deployment. One method is to place the code in an end-node initially and then the code is passively propagated along the path that a packet or capsule travels from the origin to another end (destination). In this case the code is always pulled from "previous node" whenever a node on the path is demanding for it. The method is used in ANTS. Another way is to specify a well-know node as code supplier. Once a routine is demanded by a node in the network, the node can obtain its supplier's address from the packets arrived and pull the code into the local system from the well-known supplier. Several existing systems employ this method, such as M0 and SwitchWare. Therefore, the second question is that if the two methods are optional in the system design, which one can achieve good performance and low cost? In this section we will propose our technique to answer the two questions.

Suppose that an active network consists of $N$ active nodes numbered in $1, 2, ...N$, and its network topology is given. We define a *Distance-Matrix* as $M_D$ with dimension $N \times N$. The elements of $M_D$ are denoted as $d_{ij}$, $i, j = 1, 2, ..., N$. $d_{ij}$ represents the distance, measured in the number of hops, from node $i$ to node $j$ with the shortest path between them. For each node $i$, we can also define a *Distance-Vector* as $\vec{V_i} = [N_1^i, N_2^i, ..., N_{m(i)}^i]$, where $N_d^i$, $d = 1, 2, ..., m(i)$, is the number of nodes which has the same distance, $d$, to node $i$. For example, $N_3^2 = 4$ represents that there are 4 nodes having the same distance (3 hops) to node 2. Thus, if we denote $M = max\{m(i), i = 1, 2, ...N\}$, we can list all $\vec{V_i}$ as a

matrix, $M_V$, with dimension $N \times M$. Therefore, once we get a network topology, we can specify the two matrices, $M_D$ and $M_V$, for it.

To answer the first question, we must find out at least one node in the network which has the minimal average distance to all other nodes. At first we can compute the average distance $\bar{D}_i$ from each node $i$ to all others $j$ based on the specified matrix $M_V$ by

$$\bar{D}_i = \frac{\sum_{d=1}^{m(i)} d \cdot N_d^i}{N-1} \tag{7}$$

where we assume that deploying active codes from node $i$ to any other node j has same probability. Then, we may further denote the minimum of $\bar{D}_i$ as $\bar{D}_{min} = min\{\bar{D}_i, i = 1, 2, ..., N\}$. If a node has the minimum $\bar{D}_{min}$ within the network, it can be chosen as the code supplier for the given network.

To answer the second question we need to define the average shortest-path length $\bar{H}_{\min}$ of all possible pairs of nodes in the network. This can be expressed as

$$\bar{H}_{\min} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij}}{N(N-1)} \tag{8}$$

It notes that $\bar{D}_{min}$ and $\bar{H}_{\min}$ are two different concepts. $\bar{D}_{min}$ is a measure around individual nodes for finding the best location of code supplier. Alternatively, $\bar{H}_{\min}$ is a measure about the path length between two nodes.

For the first method as described before, the code distribution takes place along the path between two nodes and thus it is reasonably to use $\bar{H}_{\min}$to represent its average cost for the system. Here we assume that the code distribution to be applied to all possible pairs of nodes with same probability. For the second method, we have to choose an optimal location for its code supplier. As discussed before, this can be found by evaluating $\bar{D}_{min}$. As a result, a procedure for optimal selection between the two methods can be described as the following:

*Step* 1 : *Specify* $M_D$ *and* $M_V$ *based on a given network topology.*

*Step* 2 : *Compute* $\bar{D}_{\min}$ *and* $\bar{H}_{\min}$ *for the network.*

*Step* 3 : *If* $\bar{D}_{\min} > \bar{H}_{\min}$, *choose the first method, otherwise, choose the second method.*

A numerical example for this decision procedure is ignored due to the size of paper restricted.

## 5. Numerical Results

In this section we show some numerical results for evaluating the performance and reliability of code distribution systems based on the model proposed in Section 3 at first and then, we will demonstrate how to make an optimal decision during the system design.

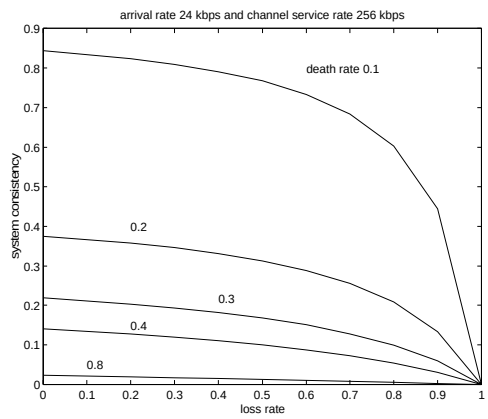


Figure 4

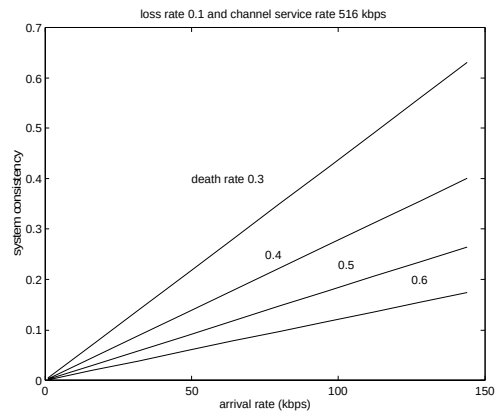


Figure 5

At beginning, we specify service rate $\mu = 256$ *kbps*, average arrival rate of original packets $\lambda = 24$ *kbps* and assume $D_1 = D_2 = D_3 = \frac{1}{\mu}$ . Traffic arrival rates and system service rate are measured in *bits/second* rather than *packets/second*. Thus the conversion between them is applied in the computation. Figure 4 illustrates system consistency $E[c(t)]$ versus packet loss rate $\delta$ with different death rate $\vec{\theta} = [\theta_1 = 0.1, \theta_2 = 0.2, \theta_3 = 0.3, \theta_4 = 0.4, \theta_5 = 0.8]$. It shows that system consistency strongly depends on packet loss rate and death rate under the given condition of service rate and arrival rate. If the system is with both low loss rate and low death rate, it can achieve high system consistency. Figure 5 demonstrates system consistency $E[c(t)]$ versus arrival rate $\lambda$, where service rate $\mu$ is set to 516 *kbps* and loss rate $\delta = 0.1$. It is noted that if $\lambda$ is increasing, $E[c(t)]$ is increasing as well. This means that code delivery is better to be performed at higher transmission rate. Figure

6 and 7 investigate system average latency $\bar{R}$ versus arrival rate $\lambda$ and service rate $\mu$ respectively. $\bar{R}$ can be increased rapidly if the arrival rate is increased and its death rate $\theta$ is smaller as shown in Figure 6. Figure 7 demonstrates that if a high service rate is provided by the system, the latency can be reduced. In summary of this evaluation, a code distribution system can achieve a good performance and reliability generally if the system has low loss rate, low death rate and higher service rate comparing with arrival rate.

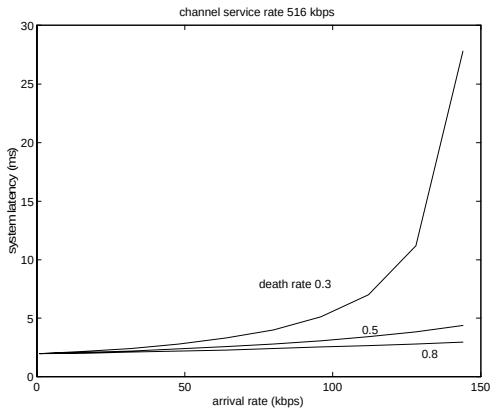


Figure 6

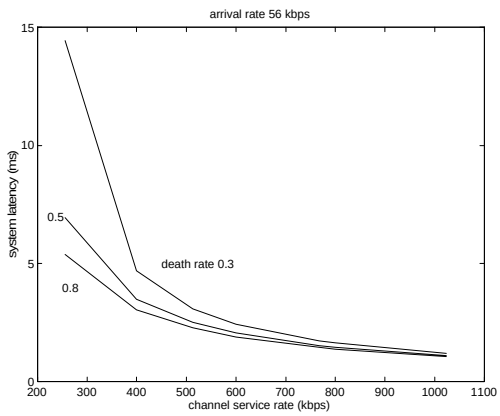


Figure 7

## Conclusions

In order to carry out the performance and reliability evaluation for code distribution systems, we have proposed our analytical model based on the queueing theory. The numerical results of using the model show how a number of factors, such as code transmission rate, network channel bandwidth, loss rate and death rate, impact on the system behaviour. In order for choosing a good code distribution method during the system design, a technique is presented to assist in making the decision between different schemes. The work is original in this topic. Active networking is an exciting subject in the area of networks and communications. There are still arguments about this new concept. Our further work will target on the design and analysis of new algorithms and protocols for code distribution in the context of the development of active networks so as to make this concept practical and potential in the future.

## Acknowledgment

We would like to thank the Wong Kuan-Cheng Research Foundation for supporting this research.

## References

[1]. David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wethrall, and Gary J. Minden, "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, pp. 80-86, Jan. 1997.

[2]. D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: A toolkit for building dynamically deploying network protocols", IEEE OpenArch 98, San Francisco, April 1998.

[3]. D. Scott Alexander, William A. Arbaugh, Michael W. Hicks, Pankaj Kakkar, Angelos D. Kermytis, Jonathan T. Moore, Carl A. Gunter, Scott M. Nettles, and Jonatha M.Smith, "The SwitchWare Active Network Architecture", IEEE Network, 12(3):29-36, May/June 1998.

[4]. C. Tschudin, "The massager environment M0 - a condensed description", Mobile Object Systems - Towards the Programmable Internet, LNCS 1222, pp.149-156. Springer April 1997.

[5]. Beverly Schwartz, Wenyi Zhou, Alden Jackson, W. Timothy Strayer, Dennis Rockwell, and Craig Partridge, "Smart Packets for Active Networks", BBN Technologies,

http://www.nettech.bbn.com/smtpkts/smart.ps.gzB.

[6]. Braden et al., "ASP EE: An Active Network Execution Environment for Network Control Protocols", http://www.isi.edu/active-signal/ARP/index.html

[7]. Baskett, F., Chandy, M., Muntz, R., and Palacios, F., "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", Journal of Association for Computing Machinary 22, 2(1975), 248-260.

[8]. Thomas G. Robertazzi, "Computer Networks and Systems: Queueing Theory and Performance Evaluation", Springer-Verlag, 1990.

[9]. Edward D. Lazowska et al., "Quantitative System Performance - Computer System Analysis Using Queueing Network Models", Prentic-Hall, 1984.