



# Efficient soft QoS guarantee in mobile ad hoc networks

Gaogang Xie, Zhenyu Li, Valérie Issarny, Alberto Conte

► **To cite this version:**

Gaogang Xie, Zhenyu Li, Valérie Issarny, Alberto Conte. Efficient soft QoS guarantee in mobile ad hoc networks. 20th International Conference on Parallel and Distributed Computing Systems: ISCA PDCS 2007, 2007, Las Vegas, Nevada, United States. pp.120-126, 2007. <inria-00415922>

**HAL Id: inria-00415922**

**<https://hal.inria.fr/inria-00415922>**

Submitted on 11 Sep 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Efficient Soft QoS Guarantee in Mobile Ad Hoc Networks

Gaogang Xie<sup>1,2,3</sup>, Zhenyu Li<sup>1</sup>, Jianing Chen<sup>1</sup>, Valerie ISSARNY<sup>2</sup>, Alberto CONTE<sup>3</sup>

<sup>1</sup>Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080, China

<sup>2</sup>INRIA-Rocquencourt, Domaine de Voluceau, 78153 Le Chesnay, France

<sup>3</sup>Alcatel Research & Innovation Center, Marcoussis, France

{xie,zyli,jianing}@ict.ac.cn, Valerie.issarny@inria.fr, alberto.conte@alcatel.fr

## Abstract

*More and more Quality of Service (QoS) sensitive applications, such as streaming media, high bandwidth content distribution and VoIP, will be deployed in Mobile Ad hoc Networks (MANETs) as part of the pervasive computing realization. However, traditional QoS guarantee technologies cannot be used directly in MANETs due to the dynamic network environment. This paper proposes a QoS management mechanism combining Caching and Backup Service Paths (CBSP) and an Enhanced CBSP (ECBSP) for soft QoS guarantee in MANETs. In CBSP, Service Provider Nodes (SPNs) with distinct Service Paths (SPs) providing the required service are found in the MANET during the service discovery phase. The client node then selects one SP to get the service and the other SPs as Backup Service Paths (BSPs). If the serving SP fails to serve the client node, the client node can handover quickly to a BSP and consume the resource in its cache during the handover operation in order to avoid service interruption. In ECBSP, the required data are further divided into several segments and transmitted concurrently to the client node through different SPs for enhanced availability. Simulation experiments in ns2 show that CBSP/ECBSP can improve the performance of applications in MANETs effectively.*

**Key words:** Mobile Ad hoc Network, QoS Guarantee Mechanism, Caching and Backup Service Paths

## I. INTRODUCTION

Mobile Ad hoc networks (MANETs) are autonomously structured with multi-hop wireless links without the aid of infrastructure network. The rapid progress in wireless communication, mobile communication, portable computers and multimedia technologies makes MANETs to be used not only for military rescue scenarios but also for industrial and commercial applications to realize the future pervasive computing environment [1,2]. With the expanding range of applications of MANETs, more and more Quality of Service (QoS) sensitive applications, such as streaming media, high bandwidth content distribution, and VoIP, will be or have been deployed in MANETs. The need for supporting QoS in MANETs is in particular becoming a key requirement [3,4,5,6,7,8]. However, the differences between MANETs and the traditional wired Internet infrastructures, such as

unpredictable link properties, node mobility and route maintenance, introduce some unique issues and difficulties for supporting QoS in MANET environments [3].

There have been mass forces focusing on improving the performance of MANETs. This has led to great achievement in recent years including network architectures [9], MAC mechanism [10,11], QoS routing protocols [12,13,14], performance estimation and improvement on TCP/UDP [15,16]. The Differentiated Service (DiffServ) [17] and Integrated Service (InteServ) [18] QoS mechanisms have been modified and used in MANETs too [20]. These QoS guarantee mechanisms improve protocol Layers 2 to 4 and require different protocols implementation to be deployed on nodes. Actually, Diffserv and InteServ are not effective for MANETs. In MANETs, resources are distributed on nodes and located by broadcast. When the service path breaks, the service has to be interrupted while the node relocates the resource. DiffServ cannot provide any mechanism to shorten the time spent in resource relocating. In InteServ, nodes try to set up a QoS connection with RSVP before data transmission. But when the connection gets interrupted during the data transmission caused by nodes departure, nodes spend lots of time to set up a new connection. QoS-sensitive applications cannot tolerate the delay and jitter caused by resource relocating and connection re-setup. In addition, some of these mechanisms are difficult to deploy since, in these mechanisms, we have to update the protocol stacks at all nodes to make them interoperate. It seems more appropriate to implement QoS guarantee at the application level. In this way, we can just provide some APIs for the development of QoS-sensitive applications development without modification of existing network protocol implementations.

This paper introduces a soft QoS guarantee mechanism by combining caching and backup service mechanisms. The client node sets up a list of Backup Service Paths (BSPs) during the resource discovery phase and Service Provider Node (SPN) transmit data to the client with source routing. The BSPs can be updated according to the status of the network. Also, resources can be fragmented into segments and transmitted concurrently from several different SPNs. Cache in the client node is used to store the resource data. With this mechanism, the MANET can provide seamless continuous service without user

awareness even when the serving Service Path (SP) is interrupted. Simulation experiments show that CBSP/ECBSP can improve the QoS of applications in MANETs.

This paper is organized as follows. Section 2 describes the service model of the MANET and defines some notations. In Section 3, the Caching and Backup Service Paths (CBSP) QoS Guarantee mechanism is introduced and the performance of CBSP is analyzed. We describe an Enhanced CBSP mechanism (ECBSP) in Section 4. Section 5 simulates CBSP/ECBSP in ns2 and the experiment data are analyzed. Finally, we conclude and discuss our future work in Section 6.

## II. MODEL AND NOTATIONS

A MANET can be considered as a direct graph  $G = \{V, E\}$ , where  $V = \{v_1, v_2, \dots, v_N\}$  is the set of nodes in the network and  $E = \{e_{i,j} \mid i, j = 1, 2, \dots, N, i \neq j\}$  define the connection relationship between nodes.  $e_{i,j} = 1$  means that  $v_j$  can send data to  $v_i$  directly, where we assume that if  $e_{i,j} = 1$ , then  $e_{j,i} = 1$ . Otherwise,  $v_i$  may receive data from  $v_j$  indirectly along multi-hops. Data can be transmitted from  $v_j$  to  $v_i$  following a path  $x$  named service path  $sp_{i,j}[x]$  and we denote the set of service paths as  $SP_{i,j} = \{sp_{i,j}[x]\}$  since there may be several paths from  $v_j$  to  $v_i$ , where  $sp_{i,j}[x] = \{v_i \dots v_k, v_s \dots v_j \mid e_{k,s} = 1\}$ .  $v_j$  can transmit data to  $v_i$  only if  $SP_{i,j} \neq \emptyset$ . If  $\exists v_m \neq v_i, v_m \in sp_{i,j}[x] \wedge v_m \in sp_{i,k}[y]$ , we say  $sp_{i,j}[x]$  and  $sp_{i,k}[y]$  are *correlative*. That is to say, the failure of  $sp_{i,j}[x]$  and  $sp_{i,j}[y]$  are dependent. The path  $sp_{i,j}[x]$  that provides the best QoS for transmitting data from  $v_j$  to  $v_i$  among the SPs is denoted as  $sp_{i,j}$ .

Other notations are defined as follows for convenience of expression.

- $p_{i,j}$ : The probability that  $v_j$  cannot provide service to  $v_i$  along  $sp_{i,j}$  due to low quality of service or path failure.
- $tl_i$ : The time (second) that  $v_i$  spends to locate the candidate service provider  $v_j$ .
- $ts_{i,j}$ : The transmission time (second) from  $v_j$  to  $v_i$ .
- $c_i$ : The cache size (Byte) in the user memory space of  $v_i$  that is used for storing data from the service provider node(s) temporarily.
- $s_i$ : The rate of resource consumption (Bps) in  $v_i$ .
- $n_i$ : The number of SPs that  $v_i$  can find to meet its QoS requirement.
- $t_{actual}$ : The actual duration from  $v_i$  generating service requirement to current time.

- $t_{invalid}$ : The invalidation time in  $t_{actual}$  due to the service path failure.
- $l_{exp}$ : The expected data packets (Byte) in  $t_{actual}$  without any service path failure.
- $l_{actual}$ : The actual data packets (Byte) in  $t_{actual}$ .
- $A_i$ :  $A_i$  denotes the set of node  $v_i$ 's neighbors in MANET. That is  $A_i = \{v_k \mid \forall v_k \in V, e_{i,k} = e_{k,i} = 1\}$ .

We introduce metrics  $q_t$  and  $q_d$  to characterize the QoS of applications with respect to service time and timely data delivering respectively. We define:

$$q_t = \frac{t_{invalid}}{t_{actual}} \quad (1)$$

where  $t_{invalid}$  is zero when there is no SP failure during  $t_{actual}$ . From the definition of  $q_t$ , we can see  $q_t = 0$  in an ideal network environment where QoS always meets the requirement of applications once a connection is set up and  $q_t = 1$  in a highly dynamic network environment where the application cannot run at all. Accordingly, we give the definition of  $q_d$  as formula (2) where  $l_{exp}$  is the packets transmitted during time  $t_{actual}$  in an ideal network environment and  $l_{actual}$  is the goodput during time  $t_{actual}$ .

$$q_d = \frac{l_{actual}}{l_{exp}} \quad (2)$$

Both  $q_t$  and  $q_d$  are metrics to evaluate the performance of service continuity, while from different perspectives. Our goal is to minimize  $q_t$ , while maximize  $q_d$ . Note that both metrics can be evaluated at any time during or at the end of the service session.

## III. QOS GUARANTEE USING CACHING AND BACKUP SERVICE PATHS

In a MANET, client node  $v_i$  has to broadcast the service request message to locate the sites hosting the resource data, and selects one among the available SPNs for data transmission according to some selection policies, such as geographic location, capacity of providers and etc. Once the SPN departs or the SP breaks off during the service session,  $v_i$  has to re-execute the procedure of resource location and broadcast the service request again. This may interrupt the service on  $v_i$ . It is necessary to introduce some mechanisms for maintaining service continuity despite the network's high dynamics. This section introduces such mechanisms based on the following assumptions:

- $\forall sp_{i,j}, \min(tl_{i,j} + ts_{i,j}) > \max(ts_{i,j})$ , since resource location usually takes more time than data transmission.
- after  $v_i$  finds a SPN  $v_j$  and sets up the service path  $sp_{i,j}$  meeting the required QoS, its cache is filled full before  $v_i$  consumes the resource
- The rate of resource consumption in  $v_i$  is invariable

with  $s_i$ .

- At least one  $sp_{i,j}$  can be found at any time.
- Any two  $sp_{i,j}$  in BSPs are independent. Thus the failure of service paths is independent. This may be not true in reality. However, we can take advantage of some mechanisms while selecting service paths to satisfy this assumption.

#### A. CBSP Algorithm

The CBSP QoS guarantee mechanism is described as follows.

##### 1) The resource location phase

- 1.1) Client node  $v_i$  locates the required resource using the gossip protocol [20]. Specifically,  $v_i$  broadcasts the resource request packet to all its neighbors. The neighbor nodes forward the packet to their neighbors except  $v_i$ . Once the resource is found at node  $v_j$ ,  $v_j$  informs  $v_i$  about the location of the resource and the path for getting to  $v_j$ . Then  $v_i$  adds  $v_j$  to the list of candidate SPNs, and stores corresponding service path in a Service Path List(SPL).
- 1.2)  $v_i$  selects  $v_j$  from the list of candidate SPNs with the least transmission delay as serving SPN and the corresponding service path (say  $sp_{i,j}$ ) as serving SP. Other SPNs and SPs are added into the BSPNs (Backup Service Provider Nodes) and BSPs (Backup Service Paths) list, respectively.  $v_i$  sends a service registration message along  $sp_{i,j}$  to make sure all nodes reserve the bandwidth for coming session.
- 1.3) When nodes along the serving SP receive the registration message from  $v_i$ , they record the session flow label and their adjacent nodes along the  $sp_{i,j}$ . Subsequently, when they receive packets belonging to the flow, they can deliver the packets to their adjacent node directly and quickly.
- 1.4) Once  $v_j$  receives the service request message from  $v_i$ ,  $v_j$  transmits the data resource to  $v_i$  with source routing along  $sp_{i,j}$ .
- 1.5) Then the system finishes the phase of resource location and comes to the servicing phase.

##### 2) The servicing phase

- 2.1)  $v_j$  transmits the data resource to  $v_i$  along  $sp_{i,j}$  with source routing.
- 2.2)  $v_i$  measures  $q_t$  and  $q_d$  during the connection duration. If  $v_i$  finds that the QoS, measured by  $q_t$  or  $q_d$ , of the current service cannot satisfy its requirement or the  $sp_{i,j}$  is broken,  $v_i$  sends a session release message to all nodes in  $sp_{i,j}$  and selects another service path from BSPs list. The

nodes along  $sp_{i,j}$  delete the flow record from the service registration table and release the network resource.

- 2.3) If node  $v_k$  along  $sp_{i,j}$  cannot provide service with the required QoS because  $v_k$  moves, departs or has heavy load, it finds another node  $v_m \in A_{k-1} \wedge A_{k+1}$  which may take over the position of  $v_k$  in  $sp_{i,j}$ .
- 2.4)  $v_m$  records the session label in its service registration table and informs  $v_{k-1}$ ,  $v_{k+1}$ ,  $v_i$  and  $v_j$ .  $v_{k-1}$  and  $v_{k+1}$  update the record.  $v_i$  and  $v_j$  update their BSPs list. If  $v_k$  cannot find  $v_m$ ,  $v_k$  informs  $v_i$  and  $v_j$  to stop current service and select another service path from the BSPs list.
- 2.5) If  $v_{k-1}$  or  $v_{k+1}$  finds  $v_k$  cannot work, and cannot find  $v_m$  either,  $v_{k-1}$  or  $v_{k+1}$  also informs  $v_i$  and  $v_j$  to stop current service and select another service path from the BSPs list.

##### 3) The closing phase

- 3.1)  $v_i$  or  $v_j$  sends a service end message to all nodes in the serving service path, deletes all BSPs information in memory and releases caches.
- 3.2) The nodes along SP/BSPs delete the record from their services registration table and update the bandwidth resource information.

#### B. Assessment of CBSP

We now assess QoS provision in MANET with CBSP. First, we discuss the additional load brought by CBSP. In a MANET consisting of  $n$  nodes, the average height of a multicast tree is bounded by  $O(\log n)$  [22]. Thus, the average length of SP is bounded by  $O(\log n)$ . Suppose the number of current sessions is  $w$  and there are  $b$  BSPs for every session. Hence, we obtain the average number of SP records in every node (denoted as  $r$ ) as formula (3). Since sessions number  $w$  is usually linear with  $n$ ,  $r$  increases logarithmically with the number of nodes. The number of messages is constant for SP updating and SP handover.

$$r = \frac{O(\log n) * w * (b+1)}{n} \quad (3)$$

We then analyze the performance improvement on QoS achieved by CBSP.

- Let  $c_i < \min(ts_{i,j}) * s_i$ . In this case, once  $v_i$  fails to get resource with required QoS from  $v_j$  through  $sp_{i,j}$ , the service breaks off. This is because the cache is not large enough to hold all the data consumed by the client during service path re-selection period. We take the failure probability of service path as  $p_{i,j}$ , and denote the current serving SPN for  $v_i$  as  $SPN_1$ . In the case  $sp_{i,1}$  fails with probability  $p_{i,1}$ , and if the ready

probability of the path from  $v_i$  to another BSPN (denoted as  $SP_2$ ) is  $1 - p_{i,2}$ , then  $t_{invalid}$  would be increased by  $\Delta t_{invalid}$ , which can be denoted as  $p_{i,1}(1 - p_{i,2})(ts_{i,2} - c_i / s_i)$ . By analogy, if at least one SP in the BSPs list is available,  $\Delta t_{invalid}$  can be denoted as formula (4).

$$\Delta t_{invalid} = \sum_{j=2}^{n_i} \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(ts_{i,j} - c_i / s_i) \right] \quad (4)$$

And if there is no one SP in the BSPs List is available, the service lookup time should be taken into account. Thus,  $\Delta t_{invalid}$  can be denoted as formula (5).

$$\Delta t_{invalid} = \sum_{j=2}^{n_i} \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(ts_{i,j} - c_i / s_i) \right] + \sum_{j=n_i+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(tl_{i,j} + ts_{i,j} - c_i / s_i) \right] \quad (5)$$

Taken  $TL = tl_{i,j}$ ,  $p = p_{i,j}$ ,  $TS = ts_{i,j}$ , we obtain

$$\Delta t_{invalid} = (TS - c_i / s_i)(p - p^N) + TL(p^{n_i} - p^N) \quad (6)$$

$$\lim_{N \rightarrow \infty} \Delta t_{invalid} = (TS - c_i / s_i)p + TL \times p^{n_i} \quad (7)$$

– Let  $c_i \geq \max(tl_{i,j} + ts_{i,j}) * s_i$ , then the probability of service interruption is zero. That is to say it is impossible for occurrence of service interruption and no BSP is necessary because once the serving SP breaks off  $v_i$  always has enough time to find another SP to obtain resource. During the procedure of resource locating and connection setting up,  $v_i$  can consume the resource in its cache.

– In the situation of  $\min(tl_{i,j} + ts_{i,j}) * s_i < c_i \leq \max(tl_{i,j} + ts_{i,j}) * s_i$ , suppose there are  $k$  SPs that satisfy  $c_i > (tl_{i,j} + ts_{i,j}) * s_i$ , and BSPs are selected randomly. Only if all the SPs in the backup list and all the SPs that satisfy  $c_i > (tl_{i,j} + ts_{i,j}) * s_i$  fail, the session would break off. Hence, the probability of service interruption is  $\prod_{j=1}^{n_i+N'} p_{i,j}$ , Where  $N' = \frac{k}{N} \times (N - n_i)$ .

$N'$  is the average number of SPs out of the BSPs List that satisfy  $c_i > (tl_{i,j} + ts_{i,j}) * s_i$ .  $t_{invalid}$  is increased by  $\Delta t_{invalid}$ , which can be denoted as formula (8).

$$\Delta t_{invalid} = \sum_{j=n_i+N'+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(tl_{i,j} + ts_{i,j} - c_i / s_i) \right] \quad (8)$$

Taken  $TL = tl_{i,j}$ ,  $p = p_{i,j}$ ,  $TS = ts_{i,j}$ , we obtain

$$\Delta t_{invalid} = (TL + TS - c_i / s_i)(p^{n_i+N'} - p^N) \quad (9)$$

$$\lim_{N \rightarrow \infty} \Delta t_{invalid} = (TL + TS - c_i / s_i)p^{n_i+k} \quad (10)$$

– Let  $\max(ts_{i,j}) * s_i < c_i \leq \min(tl_{i,j} + ts_{i,j}) * s_i$ , only if all the BSPs fail, the session would break off. So the probability of service interruption is  $\prod_{j=1}^{n_i} p_{i,j}$ .  $t_{invalid}$  is increased by  $\Delta t_{invalid}$ , which can be denoted as Formula (11).

$$\Delta t_{invalid} = \sum_{j=n_i+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(tl_{i,j} + ts_{i,j} - c_i / s_i) \right] \quad (11)$$

Taken  $TL = tl_{i,j}$ ,  $p = p_{i,j}$ ,  $TS = ts_{i,j}$ , we obtain

$$\Delta t_{invalid} = (TL + TS - c_i / s_i)(p^{n_i} - p^N) \quad (12)$$

$$\lim_{N \rightarrow \infty} \Delta t_{invalid} = (TL + TS - c_i / s_i)p^{n_i} \quad (13)$$

– Let  $\min(ts_{i,j}) * s_i < c_i < \max(ts_{i,j}) * s_i$ , and suppose there are  $k$  SPs that satisfy  $c_i > ts_{i,j} * s_i$ , and the SPs in the backup list are selected randomly. Only if all BSPs satisfying  $c_i > ts_{i,j} * s_i$  fail, the service breaks off.

Thus, the probability of service interruption is  $\prod_{j=1}^{N'} p_{i,j}$ ,

where  $N' = \frac{k}{N} \times n_i$ .  $N'$  is the average number of BSPs in the BSP list that satisfy  $c_i > ts_{i,j} * s_i$ .  $t_{invalid}$  is increased by  $\Delta t_{invalid}$ , which can be denoted as Formula (14).

$$\Delta t_{invalid} = \sum_{j=N'+1}^{n_i} \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(tl_{i,j} - c_i / s_i) \right] + \sum_{j=n_i+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j})(ts_{i,j} + tl_{i,j} - c_i / s_i) \right] \quad (14)$$

Taken  $TL = tl_{i,j}$ ,  $p = p_{i,j}$ ,  $TS = ts_{i,j}$ , we obtain

$$\Delta t_{invalid} = (TS - c_i / s_i)(p^{N'} - p^{n_i}) + TL(p^{n_i} - p^N) \quad (15)$$

$$\lim_{N \rightarrow \infty} \Delta t_{invalid} = (TS - c_i / s_i) \times (p^{N'} - p^{n_i}) + TL \times p^{n_i} \quad (16)$$

From above analyses, we find that the larger the cache is, the smaller  $q_t$  is, which introduces better QoS for applications. When  $c_i \geq \max(tl_{i,j} + ts_{i,j}) * s_i$  (case 2),  $q_t$  is equal to zero. However, large cache is not practical due to energy of mobile devices. Formula (7) and (16) illustrate that no matter how large  $n_i$  is,  $\Delta t_{invalid}$ , which is the increment of  $t_{invalid}$  due to the failure of SP, is larger than zero. That is to say, in cases 1) and 5), there is a service interruption when serving SP fails. Thus,  $c_i < \max(ts_{i,j}) * s_i$  is not recommended. From formula (10) and (13), the number of BSPs can be set according to the probability of service stability required. More stability of service can be obtained with more BSNs. In practice, we strongly recommend that  $\max(ts_{i,j}) * s_i \leq c_i \leq \max(tl_{i,j} + ts_{i,j}) * s_i$ .

#### IV. ENHANCED CBSP

In CBSP, if the current serving SP fails to provide service with required QoS,  $v_i$  has to look for a service provider from other BSPs and handover to it. It takes great risk for the session in  $v_i$  to be interrupted in this situation as analyzed in Section 3. Also, it is likely for some nodes to become overloaded when many services are targeted to the same nodes. This would lead to a load balance problem. Actually, in some cases, for example streaming video, discarding some frames is more acceptable than jitter from the user's point of view.

Thus, we split the whole file into several segments distributed on different nodes and clients can receive segments concurrently and independently from different independent SPs. In this way, we can balance the loads of SPs. We call this mechanism as ECBSP, which is abbreviation for Enhanced CBSP.

In ECBSP, once one of the concurrent serving SPs, which is transmitting segment  $s_x$  to client  $v_i$ , fails,  $v_i$  can search and transmit  $s_x$  from other BSPs. If  $s_x$  is still not in the cache of  $v_i$  when it is time for using the segment  $s_x$ ,  $v_i$  can just skip  $s_x$  and turn to  $s_{x+1}$  or  $v_i$  can wait for a scheduled deadline then skip it. If we denote the segment size as  $B_{seg}$ , then the number of segments that the cache can hold is  $m = \left\lfloor \frac{c_i}{B_{seg}} \right\rfloor$ .

We number the segments in the client's cache from 1 to  $m$ . Without loss of generality, suppose that  $k$  SPs fail concurrently, and the segments that they supply are  $s_{f1}, s_{f2}, \dots, s_{fk}$ , respectively, where  $1 \leq s_{f1} < s_{f2} < \dots < s_{fk} \leq m$ . We denote the service supplier of  $s_{f1}$  as  $SP_1$  and the cache size that can be used before deadline of  $s_{f1}$  as  $c_{available}$ . Note that all the other failures are hidden by the failure of  $SP_1$  because  $s_{f1}$  is the first segment that meets the deadline. We assume that none of the SPs that supply the segments whose indexes are less than  $s_{f1}$  would fail during the process of  $SP_1$ 's failure. Thus,  $c_{available}$  can be denoted as Formula (17).

$$c_{available} = \frac{s_{f1} - 1}{m} \times c_i \quad (17)$$

Note that the range of  $s_{f1}$  is from 1 to  $m$ , and  $P\{s_{f1} = k, | k \in [1, m]\} = \frac{1}{m}$ . Thus,

$$E(c_{available}) = \sum_{k=1}^m \left[ \frac{1}{m} \times \frac{k-1}{m} \times c_i \right] = \frac{(m-1)}{2m} \times c_i \approx \frac{1}{2} c_i \quad (18)$$

The role of  $c_{available}$  in ECBSP is similar to the cache size  $c_i$  in CBSP. In CBSP, when serving SPN fails, the available size is equal to  $c_i$ .

The situation which  $v_i$  waits for the segment and skips the segment after the threshold time (say  $T_{waiting}$ ) according to different cache size will be discussed.

– Let  $c_{available} < \min(ts_{i,j}) \times s_i$ , once  $SP_1$  failed, the service would break off. Thus, the probability of service interruption is  $p_{i,1}$ , and  $l_{exp}$  in Formula (2) is increased by 1, while  $l_{actual}$  is increased by 1 with probability  $(1-p_{i,1})$ .  $t_{invalid}$  in Formula (1) is increased by  $\Delta t_{invalid}$ , which is the minimum of the waiting threshold time and the lookup time.

$\Delta t_{invalid} = \min(T_{looking}, T_{waiting})$ , where

$$T_{looking} = \sum_{j=2}^m \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1-p_{i,j})(tl_{i,j} - c_{available} / s_i) \right] + \sum_{j=n_i+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1-p_{i,j})(ts_{i,j} + tl_{i,j} - c_{available} / s_i) \right] \quad (19)$$

– Let  $c_{available} \geq \max(ts_{i,j} + tl_{i,j}) * s_i$ , then the probability of service interruption is zero. That is to say both  $l_{exp}$  and  $l_{actual}$  in Formula (2) is increased by 1.

– Let  $\min(tl_{i,j} + ts_{i,j}) * s_i < c_{available} \leq \max(tl_{i,j} + ts_{i,j}) * s_i$ , suppose that there are  $k$  SPs that satisfy  $c_{available} > (tl_{i,j} + ts_{i,j}) * s_i$ , and the BSPs are selected randomly. Then, the probability of service interruption is  $\prod_{j=1}^{n_i+N'} p_{i,j}$ , where  $N' = \frac{k}{N} \times (N - n_i)$ .  $N'$  is the average number of SPs, out of the BSP List, which satisfy  $c_{available} > (tl_{i,j} + ts_{i,j}) * s_i$ . And  $l_{exp}$  in Formula (2) is increased by 1, while  $l_{actual}$  is increased by 1 with probability  $(1 - \prod_{j=1}^{n_i+N'} p_{i,j})$ .  $t_{invalid}$  in formula (1) is increased by  $\Delta t_{invalid}$ , which is the minimum of the waiting threshold time and the look up time.

$\Delta t_{invalid} = \min(T_{looking}, T_{waiting})$ , where

$$T_{looking} = \sum_{j=n_i+N'+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1-p_{i,j})(tl_{i,j} + ts_{i,j} - c_{available} / s_i) \right] \quad (20)$$

– Let  $\max(ts_{i,j}) * s_i < c_{available} \leq \min(tl_{i,j} + ts_{i,j}) * s_i$ , the probability of service interruption is  $\prod_{j=1}^{n_i} p_{i,j}$ . And  $l_{exp}$  in formula (2) is increased by 1, while the  $l_{actual}$  is increased by 1 with probability  $(1 - \prod_{j=1}^{n_i} p_{i,j})$ .  $t_{invalid}$  in formula (1) is increased by  $\Delta t_{invalid}$ , which is the minimum of the waiting threshold time and the looking up time.

$\Delta t_{invalid} = \min(T_{looking}, T_{waiting})$ , where

$$T_{looking} = \sum_{j=n_i+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1-p_{i,j})(tl_{i,j} + ts_{i,j} - c_{available} / s_i) \right] \quad (21)$$

– Let  $\min(ts_{i,j}) * s_i < c_{available} < \max(ts_{i,j}) * s_i$ , and suppose that there are  $k$  SPs that satisfy  $c_{available} > ts_{i,j} * s_i$ , and BSPs are selected randomly. Then, the probability of service interruption is  $\prod_{j=1}^{N'} p_{i,j}$ , where  $N' = \frac{k}{N} \times n_i$ .  $N'$  is the average number of BSPs in the BSP List satisfying  $c_{available} > ts_{i,j} * s_i$ . And  $l_{exp}$  in formula (2) is increased by 1, while  $l_{actual}$  is increased by 1 with probability  $(1 - \prod_{j=1}^{N'} p_{i,j})$ .  $t_{invalid}$  in formula (1) is increased by  $\Delta t_{invalid}$ , which is the minimum of the waiting threshold time and the look up time.

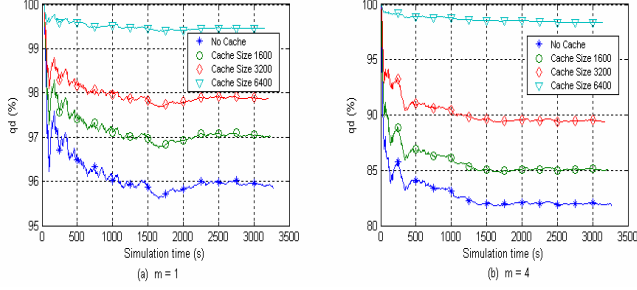


Fig.1  $q_d$  according to different cash size

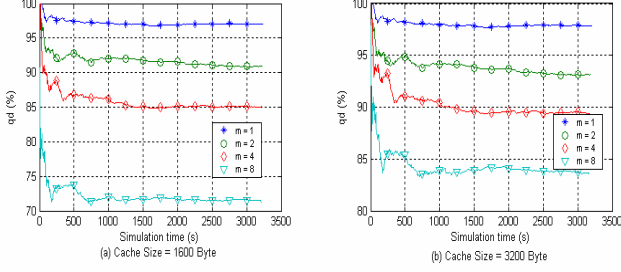


Fig.3  $q_d$  according to different m

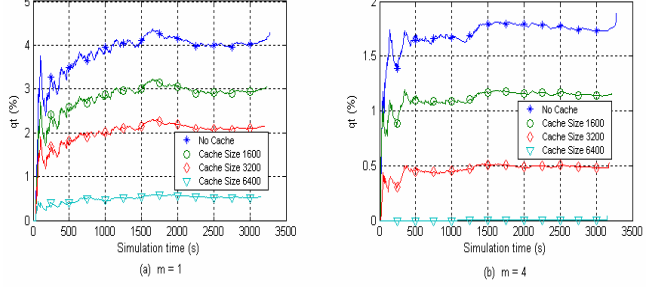


Fig.2  $q_t$  according to different cache size

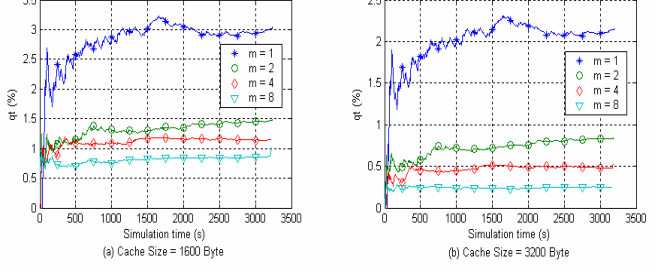


Fig.4  $q_t$  according to different m

$$\Delta t_{invalid} = \min(T_{looking}, T_{waiting}), \quad \text{where}$$

$$T_{looking} = \sum_{j=N+1}^{n_i} \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j}) (ts_{i,j} - c_{available} / s_i) \right] \quad (22)$$

$$+ \sum_{j=n_i+1}^N \left[ \prod_{k=1}^{j-1} p_{i,k} \times (1 - p_{i,j}) (tl_{i,j} + ts_{i,j} - c_{available} / s_i) \right]$$

As in CBSP, we strongly recommend that  $\max(ts_{i,j}) * s_i \leq c_{available} \leq \max(tl_{i,j} + ts_{i,j}) * s_i$  in practice. Formula (18) illustrates the expected value of  $c_{available}$  is  $\frac{1}{2}c_i$ . Thus, in ECBS, the recommended size of the client cache is  $2 \times \max(ts_{i,j}) \times s_i \leq c_i \leq 2 \times \max(tl_{i,j} + ts_{i,j}) \times s_i$ .

## V. SIMULATION EXPERIMENT

A simulation experiment has been implemented with ns2 version 2.28 to verify  $q_t$  and  $q_d$  of CBSP and ECBS. We denote  $m$  as the number of SPs for concurrent segments transmission. Thus, if  $m=1$  the CBSP mechanism works, otherwise the ECBS mechanism works.

We consider the following parameters for our simulation environment.

- The capacity and transfer time of SP are  $bandwidth=256 \text{ kbps}$  and  $delay=10 \text{ ms}$ .
- The data transfer ratio from SP to  $v_i$  is  $r1 = 128 / m \text{ kbps}$  under stable status.
- The data transfer ratio from SP to  $v_i$  is  $r2=2 * r1$  in start up procedure or when the SP recovers from malfunction.
- The time for  $v_i$  to find a BSP is  $t_l$  and the transfer time from BSP to  $v_i$  is  $t_c$ . In our experiment, We suppose  $t_l$  and  $t_c$  are all random variables, obeying respectively  $t_l \sim N(400, 100^2) \text{ ms}$  and  $t_c \sim N(400, 100^2)$

$m$ s.

- The size of  $v_i$ 's cache is  $C=0, 1600, 3200, 6400, 9600$  Byte.
- $m=1, 2, 4, 8$ .
- The file size is 50.05MB and data play ratio in  $v_i$  is 128kbps.
- The status of SPs is independent from each other and obey Poisson Process.

Table I  
TIME OF SERVICE INTERRUPTIONS (SEC)

cache(B) m	0	1600	3200	6400	9600
1	134.92	96.42	68.17	16.98	0.70
2	70.85	47.35	27.67	1.10	0
4	60.28	37.59	18.19	1.96	0.67
8	47.35	33.45	9.58	1.42	1.28

The time of service interruptions with different cache sizes are shown in Table 1. From Table 1, we can see that the time of service interruption decreases with the increase of the cache size, e.g., in CBSP from 134.92s to 0.70s and in ECBS from 41.96s to 0.67s when  $m=3$ .

The  $q_t$  and  $q_d$  of CBSP/ECBS with different parameters are shown from Fig.1 to Fig.4. Fig.1 and Fig.2 reveal that the continuity of applications is improved with the proportionally of the cache size. The values of  $q_d$  and  $q_t$  change rapidly during the session period, then come to a stable period. As showed in Fig.3 and Fig.4, the QoS has been improved with  $m$  increasing with the same cache size. There is great improvement of ECBS over CBSP, but for different  $m$  the performance is improved little in ECBS. The number of service interruptions increases with  $m$  increasing, but the total time of service interruptions decreases. That is to say,

fewer packets have been discarded or delayed. Some service interruptions become unaware and do not affect the QoS from the user's viewpoint since the lasting time is very short.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed the CBSP and an ECBSP mechanism for soft QoS guarantee in MANETs. With CBSP, some SPs providing the required service can be found in the MANET during the service discovery phase with Gossip protocol. The SPs are reported to and stored in client node  $v_i$ .  $v_i$  selects one SP to provide service with the least delay and some other SPs as backup SPs. SPN and  $v_i$  communicate with each other with source routing along the service path. When there are some changes with backup SPs, the path updated information is sent to  $v_i$  and  $v_i$  updates its backup SPs list in memory. If the serving SP fails to provide service,  $v_i$  can handover quickly to a backup SP and consumes the resource in its cache during the handover. While in ECBSP, the required data are divided into several segments and transmitted concurrently to  $v_i$  through different SPs. Comparing with other QoS guarantee mechanisms, CBSP/ECBSP utilize the information of SPNs obtained during the service discovery phase and deal with the dynamics of MANET. With these mechanisms, MANET can still provide seamless continuous service without heavy overhead when the serving Service Path (SP) is interrupted. Simulation experiment in ns2 shows CBSP/ECBSP can improve the continuity of applications in MANETs effectively.

## Acknowledgment

This research is funded by National Natural Science Foundation of China with grant no. 60403031 and 90604015, and by France Telecom R&D with grant no. 46135216.

## References

- [1] S. Corson, J. Macker, Mobile Ad hoc Networking (MANET): routing protocol performance issues and evaluation considerations, RFC 2501, January 1999
- [2] Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu and Lixia Zhang, URSA: Ubiquitous and robust access control for mobile ad hoc networks, IEEE/ACM Transactions on Networking, Vol.12, No. 6, December 2004, pp. 1049-1063
- [3] P. Mohapatra, J. Li, and C. Gui, QoS in Mobile Ad hoc Networks, IEEE Wireless Communications Magazine Special Issue on QoS in Next-Generation Wireless Multimedia Communications Systems, June 2003
- [4] Xiao Hannan, Seah WKG, Lo A, Chua K C. A flexible quality of Service model for mobile ad-hoc networks. In: Proceedings of IEEE Vehicular Technology Conf. Tokyo, Japan. 2000, 1: 445~

- [5] R. Guimarães, J. Morillo, Ll. Cerdà, J.M. Barceló, J. García. Quality of Service for Mobile Ad-hoc Networks: an Overview. Technical Report (UPC-DAC-2004-29). Polytechnic University of Catalonia, Barcelona, Spain, July 2004, <http://personals.ac.upc.edu/rpaoliel/publications.htm>
- [6] L. He, J. Musacchio, H-S W. So, and T. Tung, Adaptive Quality of Service for a Mobile Ad Hoc Network, the fifth IFIP TC6 International conference on mobile and wireless communications networks, Singapore, October 2003, <http://www.eecs.berkeley.edu/~dimakis/>
- [7] Jinshan Liu, Valerie Issarny, QoS-aware service location in mobile ad-hoc networks, in Proceedings of MDM 2004, Jan. 2004, Berkeley, CA, USA
- [8] Andrea Zanello, Daniele Miorandi, Silvano Pupolin, Paolo Raimondi, On providing soft-QoS in wireless ad hoc networks, Proceedings of WPWC'03, Oct. 19-22 2003, Yokosuka, Kanagawa, Japan
- [9] L. Chen and W. Heinzelman, Network Architecture to Support QoS in Mobile Ad Hoc Networks, Proceedings of ICME '04, June 2004
- [10] J. P. Sheu, C. H. Liu, S. L. Wu, and Y. C. Tseng, A Priority MAC Protocol to Support Real-Time Multimedia Traffic in Ad Hoc Networks, the special issue of ACM Wireless Networks, Jan. 2004 Vol. 10, No. 1, pp. 61-69
- [11] H. Zhai, J. Wang, X. Chen, and Y. Fang, Medium Access Control in Mobile Ad Hoc Networks: Challenges and Solutions, Wireless Communications and Mobile Computing, <http://www.eceel.ufl.edu/~zhai/publication/wcmc05mac.pdf>
- [12] C. R. Lin and J.S. Liu, QoS routing in ad hoc wireless networks, IEEE J. Selected Areas in Communications, 17, 8, Aug. 1999, 1426-1438.
- [13] L. Chen and W. Heinzelman, QoS-aware Routing Based on Bandwidth Estimation for Mobile Ad Hoc Networks, IEEE JSAC, Special Issue on Wireless Ad Hoc Networks, Vol. 23, No. 3, March 2005
- [14] Shigang Chen and Klara Nahrstedt, A Distributed Quality-of-Service Routing in Ad-Hoc Networks, IEEE JSAC, Special Issue on Ad-Hoc Networks, 1999.
- [15] Ruy de Oliveira and Torsten Braun, TCP in wireless mobile ad hoc networks, Technical Report IAM-02-003, University of Berne, July 2002
- [16] H. Zhai, X. Chen, and Y. Fang, How Well Can the IEEE 802.11 Wireless LAN Support Quality of Service?, IEEE Transaction on Wireless Communications, 2004
- [17] S. Blake, An architecture for differentiated services, IETF RFC2475, December 1998
- [18] R. Braden, D. Clark, and S. Shenker, Integrated service in the Internet architecture: an overview, IETF RFC 1663, June 1994
- [19] D. A. Tran, K. A. Hua, and T. T. Do, A peer-to-peer architecture for media streaming, in IEEE J. Select. Area in Comm., Vol. 22, Jan. 2004.
- [20] Xuefei Li, Lautie Cuthbert, Multipath QoS Routing of Supporting DiffServ in Mobile Ad Hoc Networks, SNPD 2005, Maryland, USA, 2005