Regular Paper

Collaboration Mechanism for Mobile IP-based Link Aggregation System

Hiroshi Mineno,[†] Yuki Kawashima,[††,☆] Susumu Ishihara[†††] and Tadanori Mizuno[†]

A variety of access technologies have been deployed over the last few years, and mobile nodes now have multiple network interfaces. In addition, mobile communications has been an active research area for several years. We have developed a link aggregation system with Mobile IP. This link aggregation system, called SHAKE, is intended to improve the throughput and reliability of wireless communication by making it possible to use multiple Internet-linked mobile nodes simultaneously and disperse the traffic between mobile nodes and a correspondent node. This paper describes a collaboration mechanism that provides packet-based and flow-based traffic distributive control based on user preferences and the cooperating nodes' states. Evaluations of a prototype implementation show that the mechanism is advantageous for a cooperative communication system like SHAKE. The results of experiments reveal that the flow-based traffic distribution is effective when the delay jitters of the nodes are extremely different.

1. Introduction

With the widespread deployment of 2G and 3G cellular systems came the notion of always being connected. Along with advances in cellular systems, a number of other access technologies have emerged and become part of our lives. Wireless LANs like IEEE 802.11a/b/g deployed in hot-spot areas provide Internet access in offices, airports, hotels, and conference rooms. In an environment of heterogeneous network access technologies, mobile users commonly make use of multiple communication interfaces such as wired LANs, wireless LANs, IrDA, Bluetooth, and cellular phones, which enable communication via the most suitable interface and access technology at all times according to the situation. The concepts and technologies for wireless networks beyond 3G are outlined in Ref. 1).

With a view towards developing ubiquitous networking, we have already proposed *SHAKE*[2], which is a scheme for sharing multiple wireless links in a cluster network environment. In SHAKE, mobile nodes connected by a fast short-range wireless link use multiple long-range wireless links co-owned by all nodes to communicate with nodes on the Internet. We consider this type of ad hoc network to be a cluster network for sharing the wireless communication links to the Internet. Schemes like SHAKE can use a function that provides an incentive for mobile nodes to cooperate and act honestly[3),4)]. Although bandwidth availability is increasing and costs are decreasing, network traffic is increasing at the same pace. To achieve high-speed data transmission, reliable communication, and high connectivity, high-speed short-range and middle-speed long-range wireless links can be combined in a heterogeneous network environment of ubiquitous networking.

A lot of research has suggested that mobile nodes should use multiple network interfaces simultaneously or dynamically select the most suitable network interfaces according to the network environment. Zhao, et al. described a way to provide global connectivity for a mobile node by using an array of network interfaces[5)]. They also showed that a flow-to-interface binding extension to a Mobile IPv4 is a useful and effective technique. R. Wakikawa et al. described a flow-to-interface binding policy for Mobile IPv6[6)], which achieves more efficient policy-based routing by maintaining a policy scheme at each mobile node and at the home agent (HA). However, it only allows versatile Internet connectivity for a mobile node that has multiple network interfaces or a network interface reconfigurable by software-defined radio technology[7)]. Because of the cost and power con-

† Faculty of Informatics, Shizuoka University
†† Graduate School of Informatics, Shizuoka University
††† Faculty of Engineering, Shizuoka University
☆ Presently with Mitsubishi Electric Corp.

sumption, it is not realistic for a mobile node to have many network interfaces or to reconfigure the interface characteristics for each flow. In the future ubiquitous networking environment, it would be realistic for the mobile node to have at most two wireless network interfaces that are reconfigured according to the situation, where one is for short-range high-speed communication with neighboring nodes and the other for long-range middle-speed communication with Internet nodes. We firmly believe SHAKE to be a realistic solution towards achieving mobile multimedia communications in ubiquitous networking.

Our research group has also developed and evaluated Mobile IP SHAKE[8]. Mobile IP SHAKE is implemented by enhancing mobile IPv4 HA so that it can disperse and tunnel the downlink IP packets to the registered care of addresses within the cluster. Although the HA selects a path so that the packet arrival time at the destination node is likely to be the earliest, it is difficult to ensure that all packets are delivered in order at the receiver. As was explained in Ref. 8), TCP degrades the end-to-end throughput because the packets are sent down paths that may have extremely various delay jitter conditions and thus may arrive out of order.

Although the above research has provided many useful mechanisms, we still lack a mechanism for ascertaining the condition of the mobile node and user preference at any given time. Even though the throughput is important for users within the cluster for downloading huge files, owners would naturally want to be given priority to utilize their own network resource. Hence, a mobile node that processes heavy jobs or does not have enough battery power would not "want" to relay traffic to others. In this paper, we describe a collaboration mechanism for Mobile IP SHAKE, which provides packet-based and flow-based traffic distribution according to the policy description and node states. The rest of paper is organized as follows. Section 2 outlines Mobile IP SHAKE and the necessity of policy-based Mobile IP SHAKE. Section 3 describes the features of Policy-based Mobile IP SHAKE in more detail. Section 4 covers the implementation architecture. Section 5 presents the evaluation results. Section 6 dicusses security and the relationship between nodes within a cluster. Section 7 briefly summarizes this work and offers some concluding remarks.

2. Mobile IP SHAKE

2.1 Overview

An example of communicating by using Mobile IP SHAKE is shown in **Fig. 1**, which illustrates a case in which three mobile nodes (MN1, MN2 and MN3) form a cluster network and share the available network resources for communicating with a correspondent node (CN) on the Internet. Each MN has two network interfaces, one is for short-range local area communications within the cluster network, the other is for long-range wide area communications with the Internet. Neighboring MNs can communicate with each other through their private IP addresses (PAs) that are assigned to the interface for the local communications. Additionally, to support seamless global roaming, we adopt Mobile IP with the co-located address mode for the interface. A global IP address, home address (HoA) or care-of-address (CoA), is assigned to the other interface for the global communications with the nodes on the Internet when MN connects like dial-up to the Internet.

We consider that a cluster network is an ad hoc network that is set up for sharing wireless communication links to the Internet. If the sum of theoretical bit rates[☆] to the Internet from the cluster network is higher than that for the communication within the cluster network and each MN uses a different long-range wireless communication channel, the cluster network achieves higher bit rate communication with the nodes on the Internet as well as higher connectivity[☆☆]. If MN1 broadcasts an *alliance request* to its neighbors and the neighboring

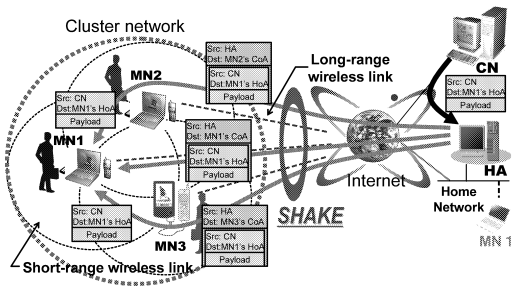

Fig. 1 Communication using Mobile IP SHAKE.

☆ The theoretical bit rate is theoretical maximum bit rate that is defined by the access technology.

☆☆ We recommend that each MN uses a different long-range wireless communication channel because interference occurs if a wireless shared link is used.

node (only MN2) replies with an *alliance response*, the cluster network consists of MN1 and MN2. Because MN3 does not reply with an *alliance response*, MN3 does not belong to the cluster network even though it is within MN1's communication range. This means MN3 is one of the nodes outside the cluster network. In fact, if nodes do not want to offer communication resources to the AL, they do not reply with an alliance response.

We call a node like MN1 that requests cooperated communication an *alliance leader* (AL), and one like MN2 that offers communication resources to the AL an *alliance member* (AM). Accordingly, the cluster network is dynamically set up by the AL and consists of the AL and neighboring AMs. Every MN can be an AL when the node is a transmitter. Therefore, there might be multiple cluster networks for each AL even though the network topology does not change.

Although an aggregating-paths protocol like SHAKE can be implemented in the application layer, doing so requires that the CN knows the multiple paths to a cluster network. This requires additional software, and it is not realistic to assume that the CN of MNs will support such a function. If a route optimization function is not used, the Mobile IP uses an HA to deliver IP packets to the MN. We modified the HA so that it can disperse and tunnel the downlink IP packets to the registered multiple CoAs within the cluster network. In this way, the CN does not have to know the multiple paths between the Internet and the cluster. Even if the CN does not know the current location and the CoA of the destination MN, all IP packets to the MN's HoA are received by the HA and relayed to the MN's CoA. If the HA knows the IP addresses of MNs in the cluster to which the destination MN belongs, it can forward the IP packets to the MNs in the cluster through the multiple paths between the cluster and the Internet even if the CN does not know that the destination MN is in the cluster.

2.2 Basic Message Flow

Figure 2 shows the basic message flow of Mobile IP SHAKE when the cluster network is formed by two MNs (MN1 and MN2). MN1 acts as an AL, and MN2 acts as an AM. Each MN has two network interfaces. One is for local communication within the cluster network and is assigned a PA. The other is for global communication to the Internet and is assigned a

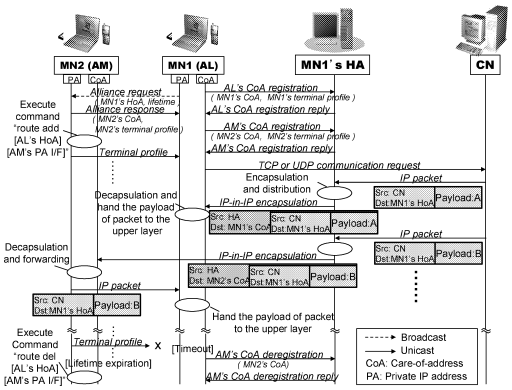

Fig. 2 Message flow of Mobile IP SHAKE.

CoA if the node moves to a foreign network. In this paper, we assume that the PA is assigned by the user and the CoA is assigned by a foreign agent deployed on the MN's visited network.

In Mobile IP SHAKE, when MN1 is away from the home network, it registers its CoA and terminal profile with its HA so that the HA knows where to forward and how to distribute its packets. The terminal profile is static information about the node, e.g., theoretical bit rate to the Internet and communication charge. If MN1 wants to use the link of neighboring MNs, MN1 broadcasts an *alliance request* to its neighbors. The *alliance request* contains the AL's HoA and lifetime for the alliance. The alliance expiration time at the AM is established by the lifetime value. If the AL wants to hold the alliance, it has to resend the *alliance request* before the expiration time is over. If the neighboring node (MN2) receives the request and agrees to it, MN2 responds with an *alliance response* that includes its CoA and terminal profile. Then MN2 executes the `route add` command to be able to relay the decapsulated IP packets that are destined for MN1's HoA. At the same time, MN1 receives an affirmative reply to an *alliance request*, after which it registers the CoA of the neighboring node and its terminal profile with MN1's HA. Therefore, MN1's HA can encapsulate an IP packet within the IP packet that destined for the AM's CoA and distribute the packet to registered CoAs. This alliance for MN1 does not need any CoA registration to the HAs of neighboring nodes, like MN2's HA in this case. When an AL receives an encapsulated IP packet from the HA, the AL decapsulates the packet and hands the payload to the upper layer. When an AM receives an encapsulated IP packet from the AL's

HA, the AM decapsulates the packet and forwards it to the AL's PA according to the routing table, which was configured when the AM sent the *alliance response* to the AL. When an AL receives an IP packet that is destined for the AL's HoA from the PA interface, the AL hands the payload of the packet to the upper layer. During the alliance, each AM periodically sends its terminal profile to the AL. This works to hold the alliance. The alliance expiration time at the AL is the received time plus four times the period of this receiving interval. When the expiration time comes, the AM information is deleted from the alliance management table. Then, the AL sends a deregistration request of the AM's CoA to the HA.

Packets sent from an AL in a cluster are distributed by the AL itself. Some of the packets are sent through the AL's own external link; as in the original Mobile IP, they are not encapsulated. The others are encapsulated and sent to the AMs, decapsulated by each AM, and forwarded through their external link. To avoid a loop within the cluster, an AL must only distribute packets from itself to its AMs.

There have been many studies on multi-hop ad-hoc networks like MANET [9]. It is possible that the MANET routing protocol is used as the routing protocol within the cluster network. However, to get the advantage of link aggregation with cluster, each MN must use different communication channel for short-range and long-range wireless communications, and the bit rate within the cluster must be higher than the aggregated bit rate of long-range links.

2.3 Necessity of Policy-based Traffic Control

In Mobile IP SHAKE, the traffic is distributed to all paths per packet, not per flow. For efficient use of multiple paths, it is important to distribute packets to an adequate path according to the condition of each path. Although the traffic distribution function at the HA and AL of Mobile IP SHAKE selects a path so that the packet arrival time at the destination is likely to be the earliest, it is difficult to keep all receiving packets in good order. The order of arrival of packets will be affected by the widely varying delay jitter conditions from path to path, and this in turn will cause TCP's congestion control to lower throughput; late packets are considered to be lost even when they arrive almost on time. If there are multiple flows and each one is distributed per path, any flow can take over the assigned path and decrease the number of out-of-order arrivals. This flow-to-path binding maintains better throughput under severe delay jitter conditions. Therefore, the mechanism for selecting either packet-based or flow-based should depend on the user's preference and path condition, e.g., the bit rate, loss rate, delay, and delay jitter. The users who act as the AL or AMs have their own preferences. For example, the AL may want to have its preference for good performance reflected on the traffic distribution. Even if an AM has agreed to cooperate with the AL, the AM might want to give priority to its own traffic over relaying traffic and set a limit on consumption of resources like CPU and battery power. This requires a mechanism that informs and applies user preferences to the traffic distribution functions at the HA for downlink traffic and the AL for uplink traffic.

3. Policy-based Mobile IP SHAKE

3.1 Traffic Control

To realize the policy-based traffic control described in the previous section, a policy is set up in each MN constituting an alliance in advance. There are two kinds of policy: the AL policy and the AM policy. The AL policy is a set of traffic distribution rules governing when the MN acts as an AL. The AM policy is a set of traffic relaying rules governing when the MN acts as an AM. These policies are sent to the traffic distribution function at the AL and conveyed to the HA by the AL when the alliance is constituted. At the same time, each MN also informs the AL of its static profile, e.g., CPU information, memory size, theoretical bit rate, and communication charge, as a terminal profile. Then, the AM periodically informs the AL of the node state according to the communication state and the load condition during the alliance, so that the AM state is dynamically applied to the traffic distribution. The alliance expiration time at the AL is the receive time plus twice the period of the reception interval. When the expiration time comes, the AM information is deleted from the traffic distribution functions. This means the AM has left the alliance. Herewith, the downlink and uplink traffic can be appropriately distributed according to the current condition of the mobile nodes and user preferences. The cooperative use of shared network resources by all AMs is also made possible. The details of the policies

```
# Downlink policy
# Source address      Protocol: S-Port   Method       Time: Method
133.70.169.223        TCP: 110           Own          10: Packet
133.70.169.233        TCP: 22            Own
any                   UDP: any           Throughput
any                   TCP: 80            Cost         60: Packet

# Uplink policy
# Destination address Protocol: D-Port   Method       Time: Method
133.70.169.4          TCP: 25            Cost         30: Packet
133.70.169.198        TCP: 22            Own
any                   UDP: any           Throughput
any                   TCP: 21            All          60: Packet
```

Fig. 3 An example of an AL policy.

are as follows.

3.2 AL Policy

An AL policy is a set of traffic distribution rules governing when a MN acts as an AL. **Figure 3** shows an example of an AL policy. The rules are generally classified into two types: those related to the downlink traffic from a CN to an AL and those related to the uplink traffic from an AL to a CN. The rules for the downlink traffic are applied at the HA. The rules for the uplink traffic are applied at the AL. Each rule contains conditional parameters and applied traffic distribution methods. The conditional parameters are the IP address of CN, the type of protocol, and the port number for the communication. There are two traffic distribution methods: the default method and the method applied after the designated period of time. The latter is invoked only when required. The period is set to be a number of seconds. These conditional parameters show a flow, so the rule can be described per flow. Because these rules are referred to from the top, the first-met rule is applied to the flow. Users can describe the preferences as the AL policy, for instance, that the flow for SSH is assigned to the flow-based traffic distribution method and the other flows are assigned to the packet-based traffic distribution method. If the application uses a specific port number per behavior, we can describe a versatile traffic distribution control in the policy.

Traffic distribution is packet-based or flow-based. Policy-based Mobile IP SHAKE provides five methods of traffic distribution. In Fig. 3, *Packet* is packet-based traffic distribution. This method enables packet-to-path binding. *Throughput*, *Cost*, *Own*, and *All* are flow-based traffic distribution methods. These methods enable flow-to-path bindings. The flow-to-path bindings ensure better throughput under widely varying delay jitter conditions on each path. The first rule of the downlink policy in Fig. 3 shows that the flow for POP communications with a mail server, or 192.168.25.110, which uses the port number 110 of TCP, is initially assigned to its own external path. If the flow continues for 10 seconds or longer, *Packet* is applied to it. The details of each distribution method are as follows.

Packet: Mobile IP SHAKE implements only this distribution method. In this implementation, we simply assume that the path is a queue. Each packet is sent to a path i where the sum of the waiting time W_i at the transmission queue and the delay d_i is the shortest one reported by the receiver. W_i is estimated as

$$W_i = \left(\frac{S}{B_i}(L_i + 1) + \tau_i - t \right) + d_i, \qquad (1)$$

where S is packet size, B_i theoretical bit rate, L_i the current queue length, τ_i the time when the packet was sent, and t the current time. We realize that this estimation can not maximize the throughput of aggregated links. This is because the estimation uses a fixed packet size and a fixed theoretical bit rate. In practice, the IP packet size differs from packet to packet and the bit rate also changes continuously. Therefore, we plan to upgrade this estimation to achieve higher throughput by using measured available bit rate, delay, and delay jitter.

Throughput: This flow-to-path binding takes particular note of throughput. The traffic distribution function selects from among multiple paths the one for which the flow may be able to achieve the highest throughput. Path selection is performed according to the theoretical bit rate of each path, which is specified in the terminal profile when the alliance is constituted. If the number of flows is greater than the number of paths, the traffic distribution function selects a path based on a calculation of the remaining theoretical bit rate by dividing the current theoretical bit rate by the expected number of flows. If some paths have the same result, the one with the lowest value of i is selected.

In fact, the available bit rate is different from the theoretical bit rate. It is also affected by many factors. Thus, we have to study the path selection algorithm in more detail in the future.

Cost: This flow-to-path binding takes particular note of the communication charge. The traffic distribution function selects from among multiple paths the one in which the flow may be able to achieve the lowest communication

charge. The path selection is performed according to the communication charge value of each path, which is specified in the terminal profile when the alliance is constituted. If some paths have the same value, the one with the highest remaining theoretical bit rate is selected. This method does not necessarily guarantee high throughput of each flow. However, there are many charge structures, so it is difficult to codify the charge value. We need to study this method in more detail in the future.

Own: This flow-to-path binding takes particular note of using only its own external link. This method is invoked if a user desires that the flow be explicitly assigned to its own external link for secure or stable communications. As for the assigned flows, this is the same as not using SHAKE.

All: This flow-to-path binding takes particular note of reliability. The traffic distribution function makes copies of the packet, and redundant packets are assigned to multiple paths in parallel. Only the packet that arrives first at the receiver is used; other redundant packets are ignored by the upper layer, e.g., TCP or RTP. This flow-to-path binding achieves reliable communications, even though the path has widely varying delay jitter.

3.3 AM Policy

The AM policy is a set of traffic relaying parameters governing when the MN acts as an AM. **Figure 4** shows an example of an AM policy. It contains the allowed bit rate for relaying traffic, the allowed relaying traffic types, and the threshold levels to reject relay traffic. Each AM applies the allowed bit rate to the communication between the AM and the AL using a queuing method. Class-based queuing (CBQ) can be used to give priority to the local traffic over relayed traffic and guarantee a certain relay rate at the AM. CBQ is a QoS scheme that identifies different types of traffic and queues the traffic according to predefined parameters.

```
# Traffic relaying policy
# Allowed bit rate for relaying traffic
128 kbps

# Allowed relaying traffic types
# Protocol: S-Port
TCP: 80

# Threshold levels to reject the relaying traffic (%)
CPU: 80          Memory: 90          Remaining battery: 20
```

Fig. 4 An example of an AM policy.

To apply the AM state dynamically to the traffic distribution, the AM periodically informs the AL of the node state according to the communication state and the load condition during the alliance. In Fig. 4, the threshold levels to reject relaying traffic are set as parameters that indicate the load condition, e.g., usage rate of CPU, memory, and remaining battery power. These levels are used to classify the load condition into three states: *high*, if one of the values goes over the threshold; *low*, if none of the values goes over one quarter the value of its respective threshold: and *middle*, if the value is between *high* and *low*. The remaining battery power is considered conversely. The communication state is classified into three states; no communication, only relay traffic, and its own and relay traffic.

A node state is classified according to these load conditions and communication states as shown in **Table 1**. There are five node states: *Free*, *Relay*, *Mix*, *Busy*, and *Down*. When the CPU utilization of the node is over the threshold level given in the AM policy, the node state changes to *Busy* because the load condition is deemed to be high. If the remaining battery power is under a quarter of the threshold level, the node state is always *Down* so that the AM does not relay traffic for the AL. Each AM decides the node state and periodically informs the AL of it during the alliance. Then, the AL forwards the AM states to the traffic distribution function at the HA when one of the received states changes from its previous state. If the node state is *Busy*, the HA does not distribute the traffic to the path. If the state is *Down*, the AM information is deleted from the alliance management table in the traffic distribution functions.

This node state advertisement also works to hold the alliance. The alliance expiration time

Table 1 Node state.

	Load condition			
Communication state	Low	Middle	High	High (Remaining battery power)
No communiation	*Free*	*Mix*	*Busy*	*Down*
Only relay traffic	*Relay*	*Mix*	*Busy*	*Down*
Its own and relay traffic	*Mix*	*Mix*	*Busy*	*Down*

at the AL is the received time plus twice the period of this advertisement interval. When the expiration time comes, the AM information is deleted from the alliance management table.

3.4 Traffic Distribution Algorithm

The traffic distribution function distributes the traffic based on the rules described in the policy file. Each MN has a policy file that contains the AL policy and AM policy. If an MN concludes an alliance with the AL, it sends the AM policy to the AL with the alliance response. The MN also sends the terminal profile, which contains its static profile. Then, the AL sends these policies and terminal profiles to the HA for downlink traffic. To apply the AM state dynamically to the traffic distribution, each AM periodically informs the AL of its own node state. The AL informs the HA of the node states only when one of the node states has changed. If the rules of each policy are updated by the user, all policy information is conveyed to the traffic distribution functions along with the node state advertisement.

If there is no rule for traffic in the policy description, the packet-based traffic distribution method is chosen as a default. It distributes the packet to all paths whose node state is *Free*, *Relay*, or *Mix*, based on estimation (1). In this packet-based method, there is no difference in priority between node states. Traffic is dispersed based on the allowed bit rate for relaying and the delay time, which is periodically measured per path.

In the flow-based traffic distribution method, the distribution is primarily according to the state of the AM that has priority. *Free* has top priority, followed by *Relay*, and then *Mix*. If the state is *Busy*, the path is not selected till the state is not *Busy*. The path *Down* is not selected because it is deleted from the alliance management table. Additionally, if the traffic distribution method is *Throughput* or *Cost*, the path is selected according to the theoretical bit rate or communication charge of each path, which is specified in the terminal profile when the alliance is constituted. If the number of flows is lower than the number of external paths in the cluster, the traffic distribution function selects paths in order of priority from all available ones. If the number of flows is greater than the number of external paths, the traffic distribution function selects a path by performing a calculation of the remaining theoretical bit rate.

Let us suppose the environment has a mixture of packet-based and flow-based traffic. Since packet-based traffic can be distributed through gaps not used for flow-based traffic, the communications use all shared network resources. Even if many nodes in the cluster request a shared network resource, each node's own network resource is guaranteed. This is because each node only lends the allowed bit rate for relaying traffic and because the AM with the overload might be deleted from the alliance since the node state changes to *Busy*. This means the AL does not use SHAKE.

4. Implementation

We developed a prototype of policy-based Mobile IP SHAKE on Linux (kernel 2.4). The prototype is a modified implementation of Mobile IP SHAKE, which is based on Dynamics HUT Mobile IP [10]. In this prototype, the flow for constituting alliance was updated so that the policy file and terminal profiles were efficiently exchanged among AMs, the AL, and HA. This prototype provides packet-based and flow-based traffic distribution for downlink traffic at the HA. As yet, traffic distribution for uplink traffic at the AL has not been implemented in the prototype. To achieve policy-based traffic control with the traffic distribution function, we used netfilter/iptables [11] and divert sockets [12]. Packets filtered by netfilter/iptables are diverted to a divert port, which is provided by the divert socket. By reading from and writing to the divert socket, traffic distribution function could control arbitrary packets according to the AL policy and AM's state. To control traffic relaying according to the AM policy at each AM, we used an iproute2+tc package [13], which provides CBQ on Linux. The bit rate between the HA and AM was controlled by invoking a command `tc` at the HA's traffic distribution function. This means the bit rate between the AM and AL would be shaped to be the assigned bit rate. The load condition for deciding the node state was obtained with a command like `vmstat` for the CPU utilization and `free` for memory, and the remaining battery power was obtained from the `proc` file system of Linux.

We evaluated the performance of policy-based Mobile IP SHAKE in an ideal experimental environment. **Figure 5** outlines the network topology for the experimental network. Two MNs with two fast Ethernet interfaces were connected to each other, and they were also connected to a PC with multiple network

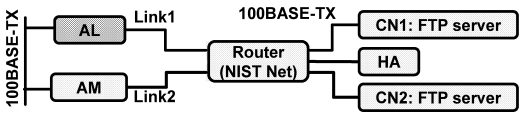

Fig. 5 Experimental network environment.

interfaces. Although the experimental network was free of wireless communications interference, it is a valid starting point for the evaluation.

The PC ran the NIST Net [14] network emulator so that it operated as a router or the Internet. One of the MNs acted as an AL, and the other as an AM. The AL's HA and two different FTP servers were connected to the different network interfaces of the router. The one-way delay of link 1 and link 2 was set to 200 msec. The packet loss rate for all links was set to 0%. The bit rate of the links between the MNs and the router was set to 384 kbps for downlink communications and 64 kbps for uplink communications. With these settings, we intended to emulate the NTT DoCoMo's FOMA service, which has a relatively long one-way delay and asymmetrical data communication speeds between reception and transmission.

5. Evaluation

5.1 Traffic Control Based on AL Policy

First, we examined the traffic control mechanism under a simple AL policy, which is shown in **Table 2** (I). This AL policy indicates that the flow-based traffic distribution method *Own* is to be applied to FTP data communications from CN1. The flow passes through only the AL's external link for up to 30 seconds. If the flow continues for 30 seconds, the packet-based distribution method is applied to the flow.

Figure 6 shows the average downlink throughput per second when the AL application layer requested a 5-Mbyte file from the CN1's FTP server with the active mode. First, the flow-based traffic distribution method *Own* was applied. The average throughput was about 381 kbps when only the AL's external link was used. Since the flow continued for 30 seconds, the traffic distribution method *Packet* was applied. The average throughput increased to 763 kbps, about twice the value of the throughput when only one link was used. That is, the AL policy was applied to the flow of FTP data communications with CN1, and the traffic distribution method was dynamically changed from flow-based to packet-based.

5.2 Traffic Control Based on AM Policy

Next, we evaluated the traffic control mechanism based on node state transition. The AL policy is shown in the Table 2 (II). This AL policy indicates that the flow-based traffic distribution method *Cost* is to be applied to the FTP data communications from CN1. The flow passes through the link according to the link's communication charge value. In the terminal profile, link 1's value was set higher than link 2's, so that link 2 would be usually selected by the traffic distribution function at the HA. The AM policy is shown in **Table 3**. This AM policy indicates that the allowed bit rate for relaying is 128 kbps, the allowed relaying traffic type is FTP communications, which is executed through TCP ports 20 and 21 for data and control messages, respectively, and the CPU threshold level to reject relay traffic is 80%. The AL requested a 10-Mbyte file from the CN1's FTP server with the active mode. The load condition of each MN was observed per second, and the node state was decided according to this load condition and conveyed to the AL at 5-second intervals.

Regarding the AM relaying traffic to the AL, the AM's CPU utilization rate was increased by running a heavy process. In **Fig. 7**, the left y-axis shows the average throughput per second at the device level for the AL and AM, and the right y-axis shows the AM's average CPU utilization rate per second. Since the total amount of uplink and downlink traffic passing through the network interface was determined by observing the `/proc/net/dev` on Linux, the throughput of each MN was calculated as the total amount of traffic passing through the two network interfaces. Thus, the throughputs in the graph show the average throughput at the device level, not at the application layer. This graph shows that the AM's throughput was about 128 kbps even though link 2's bit rate was set to 384 kbps by NIST Net. The AL's throughput (link 1) was about 0 kbps. This is because the ftp control traffic of the uplink passed through the device and the throughput was not 0 kbps. After about 285 seconds, the additional process was run on the AM. When the AM's CPU utilization rate increased to over the 80%, the AM's node state changed to *Busy*. The node state was conveyed to the AL and relayed to the traffic distribution func-

Table 2 Applied AL policy.

	Source Address	Protocol: S-Port	Method	Time: Method
I	CN1	TCP: 20	Own	30: Packet
II	CN1	TCP: 20	Cost	—
III	CN1	TCP: any	Packet / Throughput	—
IV	CN1	TCP: any	Packet / Throughput	—
	CN2	TCP: any	Packet / Throughput	—

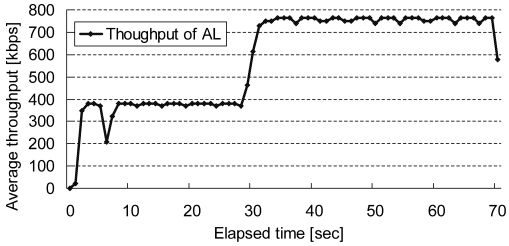

Fig. 6 Traffic control based on AL policy.

Table 3 Applied AM policy.

Parameter	Setting
Allowed bit rate for relaying	128 kbps
Allowed relay traffic type	TCP: 20, 21
CPU threshold to reject relaying	80%

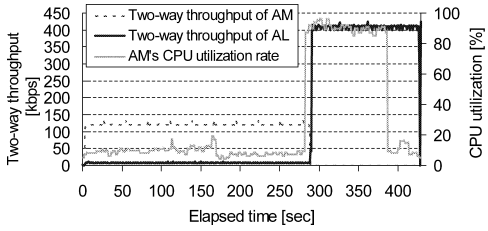

Fig. 7 Traffic control based on AM policy.

tion at the HA at the period of the advertisement interval. Traffic relayed by the AM (link 2) became nearly 0 and the AL's downlink throughput increased to the maximum bit rate, about 384 kbps. The graph shows the total amount of uplink and downlink throughput, so the throughput was above 400 kbps. This indicates that the traffic control was based on the AM's node state transition and that the AM's allowed bit rate for relaying was shaped to about 128 kbps according to the AM policy. Even though the AM's CPU utilization rate was under the threshold level of the AM policy, the flow did not return to the original path while the flow was active. This is because in the prototype specification, frequent path switching has a great impact on the other traffic.

5.3 Performance Comparison for One Flow

In this experiment, we compared the basic

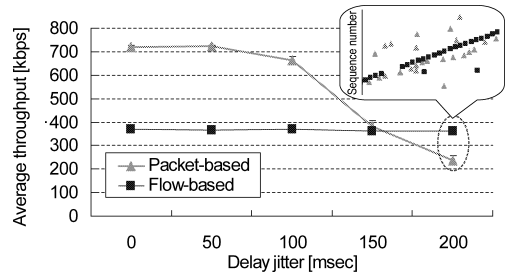

Fig. 8 Performance comparison for one flow.

performances of packet-based and flow-based traffic distribution methods under widely varying delay jitter conditions. The AL policy was as shown in Table 2 (III). The AL sent a request for a 3-Mbyte file to the CN1's FTP server in the active mode under the traffic distribution method *Packet* or *Throughput*. The average delay jitter of each link between the router and MN was set to 0, 50, 100, 150, or 200 msec, using NIST Net. All links had the same settings in that the average delay jitter would be the same as the specified value. The NIST Net generated a uniformly distributed random delay jitter between 0 and the size of the setting, i.e., 0, 100, 200, 300, or 400 msec.

Figure 8 shows the average downlink throughput at the AL application layer. In this graph, the reason the average throughput of the *Packet* method did not reach 768 kbps, which is twice the link's maximum bit rate, is that the average value was calculated by using the throughput from the beginning to the end of the communication. Since the throughput is naturally lower at the beginning of communication, the average throughput did not reach 768 kbps even though the delay jitter was 0 msec. When the delay jitter was under 150 msec, the average throughput of packet-based was higher than that of flow-based. But when the delay jitter was over 150 msec, the average throughput of packet-based was lower than that of flow-based despite that the two links were used in parallel. The upper right window shows the time transition for the TCP sequence numbers with delay jitter of 200 msec. This shows that out-

of-order packets frequently occurred when the packet-based method was used. Generally, a large difference in delay jitter per path causes out-of-order packets, especially when multiple links are used. The out-of-order packets prevent the TCP congestion window from increasing and thereby degrade TCP's performance. Therefore, when the delay jitters of the paths are extremely different, the flow-based distribution method works better than the packet-based one.

5.4 Performance Comparison for Multiple Flows

Finally, we compared the performances of packet-based and flow-based distribution methods under multiple flows. The AL sent two requests for a 3-Mbyte file to the CN1's and CN2's FTP server in the active mode, at about the same time. The AL policy was set as in Table 2 (IV). This AL policy indicates that the packet-based method *Packet* or flow-based method *Throughput* should be applied to any TCP communications from CN1 and CN2. Since each link between the MNs and the router was 384 kbps for downlink communications, the flow from CN1 passed through link 1 and the other from CN2 through link 2 according to the flow-based traffic control mechanism.

Figure 9 shows the total average downlink throughput per second at the AL application layer. At first, there was not a big difference between the packet-based and flow-based methods. As the delay jitter increased from 0 to 200 msec, the average throughput of each distribution method gradually dropped to a lower value. The throughput for the packet-based method was slightly lower than that for the flow-based one. From the results of the previous experiment, it is obvious that each throughput of flow with the packet-based method was lower than those with the flow-based one when delay jitter was 200 msec. However, if there are multiple flows or much traffic, shared communication resources are maximally utilized even when the packet-based distribution method is used. However, the TCP congestion window does not increase and this reduces the throughput. Therefore, when the delay jitters on the paths are extremely different, the flow-based distribution method works better than the packet-based one even though there are multiple flows.

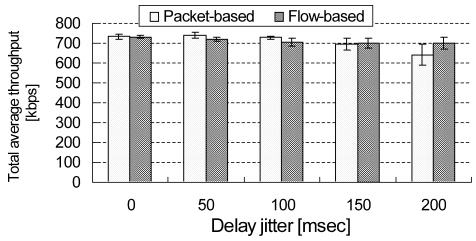

Fig. 9 Performance comparison for multiple flows.

6. Discussion

6.1 About Eavesdropping

Although AMs collaborate with the AL during the alliance, they do not necessarily have a trusting relationship with the AL. That is, AMs might eavesdrop on traffic they relay to the AL. To thwart would-be eavesdroppers, it may be possible to put an IPSec tunnel between the AL and HA. We think the IP packet can be encrypted as shown in **Fig. 10**. HA encrypts the IP packet sent by CN, and encapsulates the encrypted data, ESP (Encapsulating Security Payload) header, and trailer within IP packet that is destined for the AL's CoA if HA sends the packet to AL. If HA sends the packet to AM, the encrypted data is encapsulated within an IP packet destined for AL's HoA. This is because AM can get the destination address after decapsulation for relaying. Moreover, HA encapsulates the encapsulated IP packet within an IP packet destined AM's CoA. As mentioned in section 2.2, when AM receives an alliance request from AL, it updates it's routing table to be able to relay the decapsulated IP packets destined for AL's HoA. The double-encapsulated IP packet can be forwarded by AM through the IPSec tunnel between AL and HA.

6.2 About Falsification

The AL might also falsify the AM policy when it registers the AM policy with the HA. To prevent this from happening, the AM can be made to register a message digest of the AM policy with the HA before replying an alliance response that includes this policy. When the HA receives an AM policy from the AL, HA can check to see if the message digest is same or not. However, this requires an assumption that the HA is under the control of a third party.

6.3 About AM Accounting

The AM may be able to get into the account-

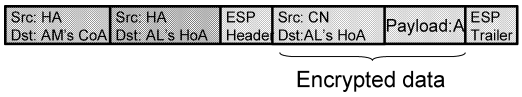

Fig. 10 Format of encypted IP packet.

ing done for relaying traffic to the AL. Thus for the AM to inform the AL of its accounting, the AM can add the accounting information within the alliance response. If AL does not want to accept the AM's accounting, it does not register the AM with HA.

6.4 About the AL's Behavior

Regarding traffic dispersion, if the AL does not inform the HA of changes in the AM state, HA will continue to deliver traffic to the AM even if the AM state is *Down* or *Busy*. To deal with such cases, we have to change the traffic distribution algorithm a little. In the prototype, the AL informs the HA of the node states only when one of these states has changed. On the other hand, the AL always relays all node states to HA in our new algorithm. The HA deletes the AM if the state is not updated within the expiration time. Thus, if the HA does not receive the state of a node, it can stop delivery to this node.

7. Conclusion and Future Research

We described policy-based Mobile IP SHAKE, which provides packet-based and flow-based traffic distribution control based on alliance leader and alliance member policies and the node states. Although we discussed the relationship among nodes within a cluster in the Section 6, there still remains some issues to be considered. It is better to assume that all nodes within a cluster maintain trusting relationships.

We evaluated the basic performance of policy-based Mobile IP SHAKE through various experiments. The results showed that the policy-based Mobile IP SHAKE works well for distributing traffic according to user preferences and the current conditions of the mobile nodes. Furthermore, it was revealed that the flow-based traffic distribution is effective when the delay jitters of the paths are extremely different. We confirmed that it is necessary to autonomously switch the distribution method between packet-based and flow-based according to the link and node conditions, and that there is a need for further evaluation in a real network environment. Moreover, we have a plan for applying this collaboration mechanism to the NEMO [15] environment.

References

1) Berezdivin, R., Breinig, R. and Topp, R.: Next-Generation Wireless Communications Concepts and Technologies, *IEEE Commun. Mag.*, Vol.40, No.3, pp.108–116 (2002).
2) Mineno, H., Ishihara, S., Ohta, K., Aono, M., Ideguchi, T. and Mizuno, T.: Multiple Paths Protocol for a Cluster Type Network, *Int. J. Commun. Syst.*, Vol.12, pp.391–403 (1999).
3) Salem, N.B., Buttyán, L., Hubaux, J.-P. and Jakobsson, M.: A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks, *MobiHoc 2003*, pp.13–24 (2003).
4) Zhong, S., Chen, J. and Yand, Y.R.: Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks, *INFOCOM*, Vol.3, pp.1987–1997 (2003).
5) Zhao, X., Castelluccia, C. and Baker, M.: Flexible Network Support for Mobile Hosts, *ACM Mobile Networks and Applications*, Vol.6, No.2, pp.137–149 (2001).
6) Wakikawa, R., Uehara, K. and Murai, J.: Multiple Network Interfaces Support by Policy-Based Routing on Mobile IPv6, *ICWN*, pp.391–403 (2002).
7) Mohebbi, B., Filho, E.C., Maestre, R., Davies, M. and Kurdahi, F.J.: A Case Study of Mapping a Software-Defined Radio (SDR) Application on a Reconfigurable DSP Core (2003).
8) Koyama, K., Ito, Y., Mineno, H. and Ishihara, S.: Performance evaluation of TCP on Mobile IP SHAKE, *IPSJ Journal*, Vol.45, No.10, pp.2270–2278 (2004).
9) Mobile Ad-hoc Networks (manet). http://www.ietf.org/html.charters/manet-ch-arter.html
10) Dynamics Mobile IP project page. http://dynamics.sourceforge.net/
11) netfilter. http://www.netfilter.org/
12) Divert Socket for Linux. http://sourceforge.net/projects/ipdivert/
13) iproute2+tc note. http://snafu.freedom.org/linux2.2/iproute-notes.html
14) NIST Net network emulator. http://snad.ncsl.nist.gov/itg/nistnet/
15) Devarapalli, V., Wakikawa, R., Petrescu, A. and Thubert, P.: Network Mobility (NEMO) Basic Support Protocol, RFC 3963 (2005).

(Received May 31, 2005)
(Accepted April 4, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.357–368.)

Hiroshi Mineno was born in 1974. He received his B.E. and M.E. degrees from Shizuoka University, Japan in 1997 and 1999, respectively. In 2006, he received a Ph.D. degree in Information Science and Electrical Engineering from Kyushu University, Japan. Between 1999 and 2002 he was a researcher in the NTT Service Integration Laboratories. In 2002, he joined the Department of Computer Science of Shizuoka University as an Assistant Professor. His research interests include mobile multimedia communications on ubiquitous networking, including wireless communication and mobility. He is a member of the IEEE, the ACM, the IEICE, and the IPSJ.

Yuki Kawashima was born in 1982. He received his B.I. and M.I. degrees from Shizuoka University, Japan in 2004 and 2006, respectively. He joined Mitsubishi Electric Corp. in 2006, and works for the information technology research & development center. His research interests include wireless communications for the support of multimedia services. He is a member of the IPSJ.

Susumu Ishihara was born in 1972. He received his BSEE and Ph.D. degrees from Nagoya University in 1994 and 1999, respectively. From 1998 to 1999, he was a special researcher of the JSPS fellowship. From 1999, he has been with Shizuoka University, where he is an associate professor in the graduate school of science and technology. His research interests are in the areas of mobile TCP/IP networking, mobile computing, and mobile ad hoc networks. He is a member of the IEEE, the ACM, the IEICE, and the IPSJ.

Tadanori Mizuno was born in 1945. He received his B.E. degree in industrial engineering from the Nagoya Institute of Technology in 1968 and his Ph.D. in computer science from Kyushu University, Japan, in 1987. In 1968, he joined Mitsubishi Electric Corp. In 1993, he became a Professor in the Faculty of Engineering, Shizuoka University, Japan. He moved to the Faculty of Informatics, Shizuoka University in 1995. His research interests include mobile computing, distributed computing, computer networks, broadcast communication and computing, and protocol engineering. He is a member of the IEEE, the ACM, the IEICE, and the IPSJ.