Service Oriented Networks – Dynamic Distributed QoS Routing Framework

See-Hwan Yoo and Chuck Yoo

Department of Computer Science and Engineering, Korea University
{shyoo, hxy}@os.korea.ac.kr

Abstract. Routing algorithms have been using a single metric such as hop count in route calculation. The selected route is optimal only in the context of the metric used. In order to support diverse service requirements from users, there is a growing need where multiple metrics need to be taken into account in routing, especially in ad-hoc routing. This paper addresses ad-hoc routing with multiple metrics. We present a new framework called service oriented network in which users can specify routing metrics and the metrics are used in route calculation. Our approach is implemented in Linux kernel and the compared with AODV. Simulation study shows that using our framework, we can improve the network lifetime by 200%.

1 Introduction

This paper is to introduce the notion of 'service' to ad-hoc routing. A service is characterized by its requirements. For example, a user may want a secure communication service even though the delay is endurably long. Another user might want low delay jitters to watch a movie on his mobile devices. Obviously, user requirements are very diverse, depending on people's preference, and also requirements may change dynamically. Some of service characteristics such as delay are network-related, and to meet the service requirements needs cooperation of the underlining routing algorithm.

However, current ad-hoc routing algorithms do not consider service requirements. For example, Ad-hoc On-Demand Distance Vector [1] (AODV) is one of the most famous ad-hoc routing protocols. It makes use of the Distance Vector algorithm, and it achieves endurable response time with low routing overhead. In other words, AODV considers only response time in route calculation. Other ad-hoc routing algorithms are similar in that a single criterion is used in route calculation.

To the best of our knowledge, this paper is one of the first attempts to incorporate the service notion into ad-hoc routing protocol. Specifically, we attempt to make ad-hoc routing flexible enough to handle user requirements so that ad-hoc routing can adapt dynamically to different characteristics of various services. This paper proposes an ad-hoc routing protocol construction framework that takes multiple criteria in route calculation to meet various

I. Chong and K. Kawahara (Eds.): ICOIN 2006, LNCS 3961, pp. 196–206, 2006.

service requirements including QoS parameters. Furthermore, our framework provides a programmable interface using simple script language to express service requirements.

This paper is organized as follows. Background and related work is described in Section 2. Section 3 highlights the core concept of the service oriented networks, the framework which constructs a service-specific network, and Section 4 presents the route construction and maintenance mechanisms in the service oriented network framework. Finally, we conclude this paper with the results in Section 5.

2 Background and Related Work

We briefly describe how ad-hoc network works. In an ad-hoc network, nodes configure themselves to communicate with their neighboring nodes. Also, any node can choose to join or leave the network, which is different from existing networks. Each node should provide its own resources to guarantee the connectivity with other nodes. Therefore, every node works as an end terminal as well as an intermediate router. In a given route, nodes included in the route ('members') should process packets. That is, all the members should involve in the route construction process, and they should maintain network information in order to provide the connectivity among themselves.

Routing protocol in an ad-hoc network provides connectivity among the nodes without preexisting infrastructure. Many routing protocols have been proposed for the ad-hoc network, and AODV and DSR are the most popular routing protocols. AODV and DSR [2] are called on-demand routing protocols because they do not keep the past network information such as topology. AODV construct a route to a designated destination based on the node's responsiveness. It selects the shortest delay path in the probing time. AODV does not maintain the route to arbitrary node in advance; instead, it probes the whole network to find a route when a new session starts. Every node in AODV network keeps routing table and it contains the next hop to destination in distributed manner. On the other hands, in DSR, only the source keeps explicit node list on the path. To keep the recent neighbor's information, AODV optionally implements a scheme that keeps sending hello messages periodically.

One of the important metric in ad-hoc routing protocol is network lifetime. Network lifetime is highly dependent on the remaining energy of the member nodes in the network. Network data transmission is one of the dominant energy consuming parts in a system, and many strategies for energy efficient management are proposed. Specifically, for longer network lifetime, several routing protocols are proposed [3]. One of them is based on total transmission power. In the scheme, route is selected based on the battery consumption of all the nodes on the path when a packet is sent. Namely, the total energy consumption for a packet on each path is calculated. If we assume that every node's energy consumption for a packet is the same, then the minimum hop route is naturally selected. However, this scheme does not consider the network lifetime. So the

lowest battery node should not use its energy on forwarding. Therefore, Toh[4] suggested a routing scheme that can avoid the lowest residual battery node. In CMMBCR[4], among all the nodes on each path, the minimum battery capacity remaining node is selected. Then, a path that has the maximum lowest energy capacity is selected as the best route. More distributed power control algorithms are proposed and power efficient protocol related work is summarized in [5].

Security issue in ad-hoc network is serious because all the nodes in the network take part in network management. For example, every node maintains a routing table and exchanges routing messages. Moreover, temporal communication failures occur often in ad-hoc networks, and it is very difficult to distinguish them from the network attack. Several studies reveal the exploitation for the routing protocol, and propose defensive schemes against security risks for ad-hoc routing protocols [6], [7].

QoS routing protocol is proposed for satisfying the QoS-related requirements [8], [9]. For realtime applications, network delay for the communication should be pre-determined. However, finding a route with delay-bound is difficult in two folds. At first, the network dynamics influences the network delay, and secondly it is proven that finding the lowest delay route to arbitrary destination is an NP-complete problem. Therefore, heuristics and approximation algorithms have been developed. Current QoS routing can be categorized as follows: 1) source routing, 2) distributed routing, 3) hierarchical routing. Various routing objectives lead different routing strategies such as bandwidth-based route selection, delay-bound route selection, etc. Most of the QoS routing protocol does not consider dynamics in the network environment and requirement change, and they solve only routing problem in static environment. Wide meaning of QoS covers various criteria such as user response or security level, which are service-specific criteria. Current QoS routing study focus on network issues such as delay bound, available bandwidth, or jitters, and the algorithms for approximation. This work enlarges the area of QoS routing in more general meaning of QoS with flexible framework.

Active network is a research area for adapting network systems to dynamically changing environment. Several approaches are proposed for supporting flexible and programmable network construction [10], [11]. Programmable network provides flexibility in network management. Because the network management is a complex task, network administrator wants it to be done automatically. However, some research work shows a possibility of active network with real experiments using Tunneling protocol [12].

3 Concept of Service Oriented Networks

The core concept of the service oriented network is presented in figure 1. The service oriented network is to support services with different network requirements so that the services should be provided in the timely manner and can be changed easily or migrated to other environment. Different services such as security or multimedia streaming service have their own requirements. In service oriented network, the middle layer, called collaboration layer, gathers the requirements

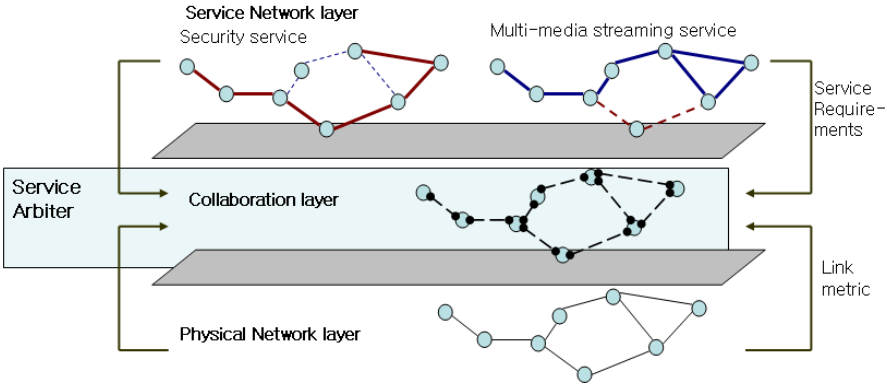

Fig. 1. Conceptual view of service oriented network

from the upper layer (user services) and constructs a service-specific network topology. The collaboration layer also gets link information, called link metric, from physical network layer. In the collaboration layer, there is a service arbiter that constructs a service specific network with the link metric information.

For example, we can imagine a following scenario for the service oriented network. A user wants to watch a movie. In the middle of the process, user authentication is required because multimedia service is a high-cost service and only valid user should access the service. Once authentication is completed, the codec for the media contents is required because codec influences the quality of the media streaming display, and it is highly dependant on the device processing power and media data size. If codec is not found in the local display device, the codec can be downloaded from the codec database. Finally, the multimedia application contents are streamed through a network. Streaming service requires realtime processing of contents and delay jitter is very sensitive factor in realtime data process.

Collaboration layer receives metrics using a simple script language. User or application service specifies his requirements, and the requirement specification is processed via service arbiter. Therefore, we can provide the logical view of the network which can support specific application service we targets on.

Although the language approach provides good flexibility, this method has a weakness. If the language is an interpretation one, we need a virtual machine or interpreter to execute the instruction and the performance will be quiet poor. If it is a compilation language, we need to compile before setting the configuration.

In both cases, there should be a structure for checking the input grammar, which is called parser in the compilation language. We develop intermediate structure for the metric calculation is developed in order to reduce the grammar check overhead. Using Lex and Yacc, we implemented software that is an intermediate code generator from the metric definition language.

Once the routing metric is specified, the service oriented network infrastructure prepares a code generator for the metric definition. Specifically, the input

string of the cost definition is parsed and makes a stack structure. When we evaluate the routing cost for the specific route, metric factors come from the symbol table. Each metric factor is previously calculated and stored in the symbol table. By interpreting the cost function, the structure has flexibility and the performance problem can be resolved with the help of calculator which is created when we parse the cost function.

4 Route Construction and Management

4.1 Routing Metrics

In service oriented network, we need a route construction rule for each routing metric. The reason why an existing ad-hoc network cannot provide the different characteristics is that all the nodes have only one routing metric. Consequently, we have to define new routing metrics that reflect service characteristics and distribute them to all the nodes in the network. In service oriented network, any node can create a new routing metric, and all the nodes should understand and calculate the proper cost function of the metric.

Service oriented network uses two features. One is dynamic routing metric processing. Application service requirement is expressed in simple script language, and it is processed via service arbiter to calculate cost for a route. It allows complex and dynamic metric calculation so that the calculation of multiple metrics involves quiet a large overhead such as parsing the metric definition and processing overhead. Using the intermediate code, it significantly reduces the overhead such as run-time parsing of requirement specification.

The second feature is two combination rules for multiple metrics. The combination rules unify multiple metrics to a new single routing metric. These rules are essentially a normalization scheme for multiple routing metrics. The first combination rule is to deal with different scales of metrics. For example, the hop count metric has a range of integer value, but remaining battery metric has a range of real number within 0 and 1. If we add two metrics, the integer (hop count) will dominate the combined metric, and the remaining battery metric is ignored. Combining multiple metrics gets more difficult when the range of a metric is not known. Two examples of hop count and remaining battery metrics are normalized as follows.

- Hop count metric
  Hop count is one of the most popular metrics. According to the combination rule, the hop count is normalized as the ratio of minimum number of hops between the route that includes the specific path and the optimal route (minimum number of hops to destination). By taking the relative value of number of hops, we can easily set the range of the metric value. Hop count metric function f can be expressed as in Eq.(1).

  $H_{dmin}(i)$ = (Minimum number of hops from node i to destination node d).
  $H_d(i,j)$ = (Minimum number of hops from node i to destination node d, including path from node i to node j).

$$F_d(i,j) = \frac{H_{dmin}(i)}{H_d(i,j)}, \tag{1}$$

where j is a neighboring node of i.

So, the combination rule gets a value between 0 and 1 for the hop count metric, and a lower value means a better route.

– Remaining battery metric
This metric selects the maximum value among remaining batteries of each node in a route. Then it is normalized as in Eq. (2), and a bigger value means a better route.

$B_d(i,j)$ = (Minimum battery remaining on the route
from node i to node d including path (i,j))
,where j is a neighboring node of i.
$B_{dmax}(i) = \underset{k}{Max}(B_d(i,k))$

$$F_d(i,j) = \frac{B_d(i,j)}{B_{dmax}(i)}, \tag{2}$$

where j is a neighboring node of i.

After metrics are normalized, we need another combination rule simple combination of the normalized metrics does not result in an optimal route. The reason is that for some metrics such as remaining battery and security level, a larger value means a better route whereas in metrics like number of hops and delay jitter, a smaller value is better.

So the second combination rule is: for the metrics where the larger value is better, Eq. (3) is used to generate the unified metric.

$$Metric\_unified_{da}(i,j) = \frac{Metric_{da}(i,j)}{Max(Metric_{da}(i,k))}, \tag{3}$$

where k is a neighboring node of i.

In Eq. (3), $metric\_unified_{da}(i,j)$ means normalized metric that returns the goodness of a route from the node i to destination node d through the path (i, j).

For the metrics where a smaller value is better, the unified metric is calculated below:

$$Metric\_unified_{da}(i,j) = \frac{Min(Metric_{da}(i,k))}{Metric_{da}(i,j)}, \tag{4}$$

where k is a neighboring node of i.

Through two combination rules, $metric_unified$ of Eq. (3) and (4) is used for route calculation with multiple metrics.

4.2 Comparison with AODV

Route Construction and Management. Although AODV constructs a network on-demand, the node always chooses the route that has the least delay time. To avoid loop in a network topology, AODV does not process more than twice for packets that have same broadcast ID. Namely, AODV processes only the first routing packet for the same broadcast, and this means that all the packets arrived later are silently ignored.

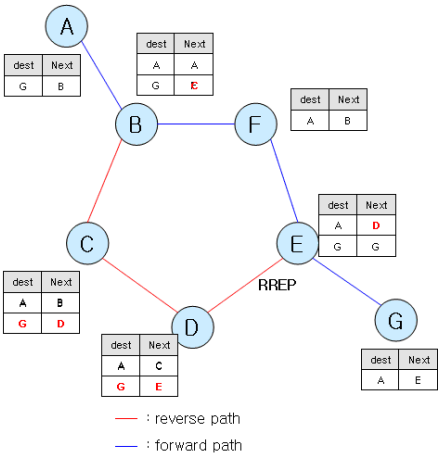

Fig. 2. Routing table update by RREP propagation

On the other hands, service oriented network should not ignore the rest of the packets. Constructing a network that takes service characteristics into acount, we have to gather the information such as security measure or remaining battery as well as delay or hop count. In AODV implementation, we consider the metric processing whenever it receives a routing message. However, we do not forward route request (RREQ) message because this message causes infinite message forwarding and loop of network topology.

To calculate each of the metric, RREQ packet carries metric definition. When a node receives a RREQ packet, the packet is forwarded as soon as possible to the next node. When the packet is being forwarded, it also carries updated metric value for each metric. In the meanwhile, intermediate code is prepared and routing metric is evaluated as to the metric value.

As a matter of fact, AODV construct a reverse path which is destined to the source node in route construction stage. Forwarded RREQ makes a reverse path and the route evaluation can be done after that all of the information is gathered. Actually, the route to the designated destination is confirmed by the RREP packet. Once the RREQ packet reaches the destination node, RREP packet is generated and returned on the reverse path which is

constructed previously. Therefore, the reverse path can be chosen after getting all of neighbor's information.

For example, as in the figure 2, let A is a source node, and we try to find a route to G, then at node D and F the RREQ packet is broadcasted. The first-comer to the E node, D or F node can create a reverse route to node A, and the RREQ from left one (F or D) is ignored. The figure presents that E chose F as next hop to the reverse route to A. RREQ broadcast from D is silently discarded and there is no table update.

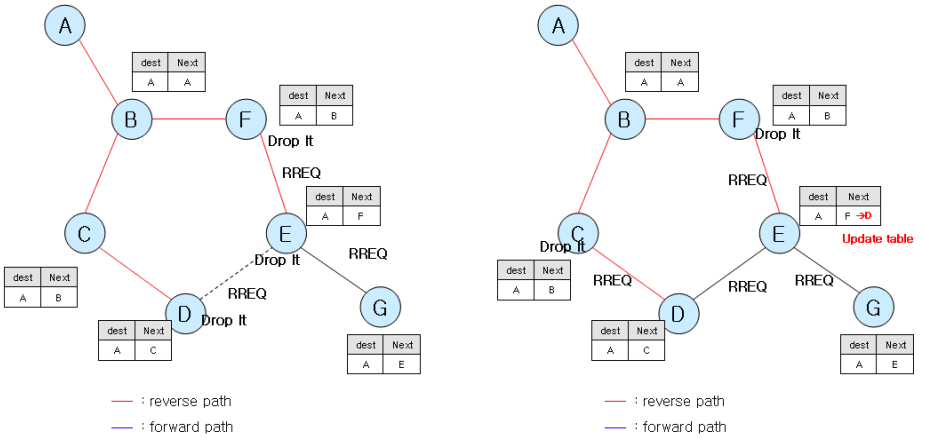

Fig. 3. Original AODV operation and AODV of service oriented network

Modified AODV is also presented in figure 3. All the packets to the node E are processed and routing table is updated. Next hop to the destination is changed to node D and in the RREP forwarding phase, E sends RREP to node D and the forwarding route is constructed as figure 3.

4.3 Simulation Results

For the performance evaluation, we have implemented the service oriented network framework on the NS-2 simulator. Simulation topology used is presented in figure 4. The following two services are used.

Metric alteration is done manually, and the metric used in simulation has chosen from the following scenarios.

- Service 1: service that has a requirement of minimum delay.
  (node 2 to node 4)
- Service 2: service to maximize the network lifetime by saving battery of nodes.
  (node 1 to node 4).

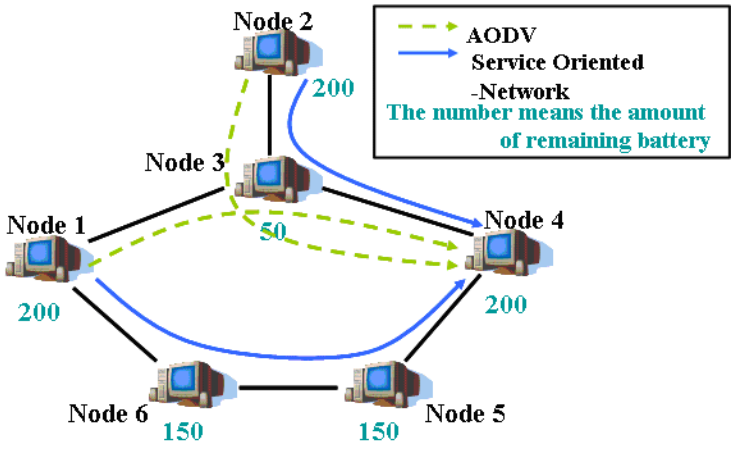

Fig. 4. Simulation scenario of the service oriented networks

Table 1. Comparison of the network life times and throughputs on session

	Network life-time(s)	Throughput on session
AODV	44	185Kbps
Our framework	127	186Kbps

For service 1, AODV calculates route: node 2 → 3 → 4, and for service 2, node 1 → 3 → 4. But service oriented network framework, because the remaining battery metric is used, the route for service 2 is node 1 → 6 → 5 → 4. The new route chosen by service oriented network framework saves battery in node 3, which extend the network lifetime. The detailed result of network lifetime is presented in table 1. The result presents that service oriented routing has similar throughput as AODV but has extended network lifetime 200% longer.

5 Conclusion

There is a growing need in ad-hoc networks for handling various and dynamic services. However, ad-hoc network construction protocols such as AODV do not consider the service characteristics. This paper develops a flexible and efficient framework for handling the dynamically changing service requirements.

Our framework has distinct features: 1) language approach to gives flexibility to specify routing metrics; 2) combination rules to unify different metrics. The simulation study reveals that incorporating service characteristics in routing is more useful for ad-hoc networks.

Acknowledgement

This research is supported by the Ubiquitous Autonomic Computing and Network project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea and partially supported by No.R01-2004-10588-0 from the Basic Research Program of the Korea Science & Engineering Foundation, and also partially supported by Korea university Grant.

References

1. C. Perkins, E. Belding-Royer and S. Das: AODV RFC 3561, Internet Engineering Task Force(IETF), July 2003.
2. David A., Maltz David B., Johnson and Yih-Chun Hu, The Dynamic Source Routing Protocol, experimental edition, July 2004. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt.
3. S. Singh, M. Woo, and C. Raghavendra, Power-aware routing in mobile ad hoc networks, In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking(Mobicom '98), pages 181-190, October, 1998.
4. C. Toh, H. Cobb, and D. Scott, Performance evaluation of battery-life-aware routing schemes for wireless ad hoc networks. In Proceedings of IEEE International Conference on Communications (ICC '01), June 2001.
5. Christine E. Jones , Krishna M. Sivalingam , Prathima Agrawal , Jyh Cheng Chen, A Survey of Energy Efficient Network Protocols for Wireless Networks, Wireless Networks, v.7 n.4, p.343-358, 2001.
6. Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth Belding-Royer, A Secure Routing Protocol for Ad hoc Networks, In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP '02), November 2002.
7. Manel Guerrero Zapata, Secure Ad hoc On-Demand Distance Vector(SAODV) Routing, Mobile Computing & Communications review, v. 6 n. 3, p. 106-107, 2002.
8. Shigang Chen and Klara Nahrstedt, An Overview of Quality-of-Service Routing for the Next Generation HighSpeed Networks: Problems and Solutions, IEEE Network, Special Issue on Transmission and Distribution of Digital Video, v. 12, n. 6, p. 64-79, 1998.
9. Z. Whang and J. Crowcroft, Quality-of-Service routing for supporting multimedia applications, IEEE Journal on Selected Areas in Communications, v. 14, n. 7, p. 1228-1234, 1996.
10. David J. Wetherall, John Guttag and David L. Tennenhouse, ANTS: Network Services Without the Red Tape, IEEE Computer, v. 3, n. 4, p. 42-48, 1999.
11. Christine E. Jones , Krishna M. A. B. Kulkarni, G. J. Minden, R. Hill,Y. Wijata, A. Gopinath, S. Sheth, F.,Wahhab, H. Pindi and A. Nagarajan, Implementation of a Prototype Active Network, In Proceedings of the 1st IEEE Conference on Open Architectures and Network Programming (OPENARCH 98), April 1998.
12. Sanjai Narain, Thanh Cheng, Brian Coan, Vikram Kau, Kirthika Parmeswaran, William Stephens. Building Autonomic Systems Via Configuration, In Proceedings of 5th IEEE Annual International Workshop on Active Middleware Services (AMS '03), p. 77, June 2003.

Appendix 1: Lex and Yacc Grammar for Cost Definition Language

Grammar definition in yacc.y

```
statement_list: statement '\n'
    | statement_list statement '\n' ;
name_list: NAME
        | name_list ',' NAME ;
statement: NAME '=' expression
    | expression
    | DEF_METRIC name_list
    | DEF_COST_FUNC '(' DEF_METRIC name_list ')'
      '{' expression '}'
    | eval_metric ;
metric: NAME '=' expression ; metric_list: metric
    | metric_list metric
    | metric_list ',' metric ;
eval_metric: DEF_METRIC '{' metric_list '}' ;
expression:
    expression '+' expression
    | expression '-' expression
    | expression '*' expression
    | expression '/' expression
    | '-' expression %prec UMINUS
    | '(' expression ')'
    | NUMBER
    | NAME
    | FUNC '(' expression ')'
    | '\n' ;
```

Lexical Analysis rule in lex.l

```
METRIC { return DEF_METRIC; } COST_FUNC {
    cost_function_index = 0;
    return DEF_COST_FUNC;
    }
([0-9]+|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?) {
    yylval.dval = (double)atof(yytext);
    return NUMBER;
    }
[ \t] ; [A-Za-z][A-Za-z0-9]* {
    struct symtab *sp = symlook(yytext);
    yylval.symp = sp;
    if (sp->funcptr)
        return FUNC;
    else
        return NAME;
    }
"$" {return 0;}
\n | . return yytext[0];
```