Regular Paper

SimRouting: A Routing Simulation Tool

Yasuhiro Ohara,[†1,*1] Masaki Minami,[†2] Osamu Nakamura[†2] and Jun Murai[†2]

A new network simulation tool called *SimRouting* is introduced in this paper. Focusing on the routing system, SimRouting targets the simulation of the summary of the entire network state on a given network setting. It enables to grasp state of a communication network in an abstract way similar to as done in Graph theory and theory of Network Flows. This is a new approach since existing network simulation tools employ the communication layer stack model, where consulting many combinations of shapes of traffic and topologies is difficult because communication protocols and traffic must be defined for each pair of nodes. Objectives of SimRouting include 1) to illustrate the resulted network state for a network setting, 2) to handle many combinations of network settings, 3) to consult the influence of a change in the network setting, 4) to evaluate routing systems by comparison, and 5) to provide an environment to develop a new routing system. The tool is evaluated by comparisons on supported functions with the other network simulation tools. It is shown to have desired properties for network simulation.

1. Introduction

Internet communication is used for many purpose in today's human life, thus the importance of the performance of the Internet communication is increasing. The state of the communication network (or *network state*) determines the performance of user communications in the network. The network state consists of many states of individual components in the network, such as the operational state of the links in the network, the link utilizations, the existence of congestion, and the degree of the communication delay. For example, congestions in the network must be avoided since they induce additional communication delay and packet loss, aggravating the communication performance. Because the performance influences all types of communication, keeping the network state good and to improve it are important both for network administrators and users of the communication network.

There are many activities currently undertaken to keep the network state good. 1) To design the network by network design tools such as Cisco MetroPlanner[6], AT&T Lab's INDT[9], WANDL's NPAT[32], RSoft's MetroWAND[26], and OPNET's IT/SP Guru Network Planner[24]. These tools help, for example, to decides the location of Point of Presences (PoPs) and the bandwidth of circuits. 2) To know the current operational state of the network by technologies such as SNMP[12], NetFlow[5] or sFlow[28]. Sometimes Network Management Systems (NMSs) such as HP Network Management Center[13] (formally known as HP OpenView), IBM Tivoli Software[14], and CiscoWorks[7] are employed to utilize these technologies efficiently. 3) To perform traffic engineering. Traffic engineering is the task of manipulating the traffic in the network to consume the network bandwidth efficiently. Traffic engineering is done by one of three ways: routing metric optimization such as OSPF cost optimization[11], BGP route control[2], and the utilization of virtual circuit technology such as MPLS with CSPF[34]. The traffic engineering by virtual circuit technology is considered the most effective, thus many tools are developed to help it. Examples of these tools include WANDL's IP/MPLSView[31] and Cariden MATE[3]. 4) To simulate a possible future network state on a what-if scenario. The discrete-event simulation tools are used for this purpose. Examples of these tools include NS-2[19], QualNet[27], OPNET Modeler[20], OmNet++[22], and J-Sim[17]. These simulation tools are revisited in Section 4.

With the goal of improvement of the network state, we insist in this paper that it is necessary to focuse on two additional objectives: 1) investigation of many network states on various network conditions, and 2) improvement of the

†1 Keio Research Institute at SFC
†2 Faculty of Environment and Information Studies, Keio University
∗1 Presently with Japan Advanced Institute of Science and Technology (JAIST)

routing system.

First, investigation of many network states on various conditions helps to improve the network state. Conditions include the network topology (i.e., the shape of network graph structure), the operational state of each network links, the setting of routing metrics, and the amount of network traffic (terminologies are defined in Section 2). With the investigation of the network state on each of the modified conditions, we can search a better setting of the network topology or of the routing metrics. Further, for the possible future occuring of the modified condition, we can understand what will happen then (i.e., predict) and prepare for possible future problems.

Second, the routing system must be improved. The routing system is very important since it decides the network path that determines the performance of the user communication, such as available bandwidth, throughput, communication delay, jitter, and reliability of packet arrival. Improvement of the routing system, including the improvement in routing algorithms and routing protocols, contributes to improve the network state. To improve the routing system, an environment to develop new routing systems and some means to evaluate routing systems are necessary.

Since none of network design tools and simulation tools is proper for the objectives (current existing tools are discussed in Section 4), a new simulation tool called *SimRouting* is developed in this paper. Approach in the SimRouting tool is novel in terms of the following.

(1) It utilizes the abstract perspective such as done in Graph theory and Network-flow theory, in contrast to the communication layer model of existing tools, so that 1) it is easy to utilize/develop algorithms in Graph theory, 2) a number of moderate size network (such as ˜1000 routers) can be easily defined (e.g., complex network traffic is easily defined as a matrix), and 3) importation of other data is trivial.

(2) Core simulation is made as simple as possible, so that the development of new algorithm in the tool requires less prerequisite knowledge about the tool. This is in contrast to the existing simulation tools which is complex with more functions and models embeded.

(3) Each network variables are defined almost independently, so that 1) many network conditions can be simulated by combinations of network variables, and 2) the investigation of the influence of a network variable is easy, by changing only one network variable between two simulations.

SimRouting developed in this paper helps users to investigate the state of the network in various combinations of network graphs, link capacities, routing algorithms, routing metrics, and network traffic demands. In SimRouting, traffic loads in the network links (i.e., link utilizations) and the reliability of packet arrival on given link reliabilities can be easily calculated on the applied routing algorithm. The support for many network conditions enables users to evaluate technologies that was difficult to evaluate in the past, such as routing algorithms, by verifying it on many number of conditions. It is expected that SimRouting contributes to obtaining the overview of the network state, indicating the insufficient part of the network that should be reinforced, predicting the network state after network changes, and development and evaluation of routing systems.

The rest of this paper is organized as follows. Section 2 defines the terminology in this paper. Section 3 revisits the need for a new simulation tool, and describes the required feature for it. Section 4 discusses the other existing tools as the related works. Section 5 describes the design and implementation of the SimRouting. Section 6 shows the example usages of SimRouting to show the feasibility of the approach. Section 7 evaluates the SimRouting, by comparison in supported features and by showing the easiness of defining various network conditions. Section 8 gives concluding summary.

2. Terminology

routing system is a system consist of the routing algorithm, the routing protocol, and the setting of the routing metric. The routing system decides the routes on the network.

network variable is a variable in communication network. It indicates any variables, including the network topology, the capacities on each links, the operational state of each network links/routers, the setting of routing metric, the identity of the employed routing algorithm, the network traffic demand matrix, the routes calculated by

the routing algorithm, and the loads, the available bandwidth, and the utilizations on each links.

network parameter is a network variable that is specified by the network administrators, the network operators, or the users. For example, the network topology and the capacities on links are network parameters specified by the network administrators. The network traffic (demand) matrix is also a network parameter, recognized as the value specified by the users collectively.

incident variable is a network variable that is modified by incidents or system events. The operational states of links and routers are network incident variables, since they can be modified by link-down and router-down incidents.

network condition is a combination of the values of the network parameters and incident variables. It acts as a parameter to determine the network state.

resulting variable is a network variable that is a result of a system, such as routing system or traffic forwarding system according to the network condition. The routes and the utilizations of links are the resulting variables resulted by the routing system and traffic forwarding system, respectively.

network state is a combination of a network condition and values of resulting variables, and is equivalent of a combination of values of all network variables. It means the integrated state of all network components. The network state is the environment that determines the communication performance.

For example, the network topology, the capacities on each links, the setting of routing metric, the identity of the employed routing algorithm, and the network traffic demand matrix are the network parameters. The operational state of each network links/routers are the incident variables, and the routes and the utilizations on each links are the resulting variables. All of them directly or indirectly relate to the communication performance, and are expressed collectively as the "network state".

3. Motivation and required features for routing simulation

This section provides the motivation and the problem description, for the two objectives of the investigation on many network conditions and the improvement in routing systems.

As mentioned earlier, investigation of the network state on many network condition contributes to search for a better setting of network topology and/or routing metrics, and also contributes to future prediction and preparation. It is imperative for us to know what would be the resulted network state when something is changed in the network (i.e., what-if scenario). For example, knowing the network state after changes are necessary, such as that a link is added, a routing metric is changed, or a traffic demand changes. Particulary, since traffic demands may change and the estimation may be erroneous, many shape of traffic demands need to be considered and prepared, in order to keep the real network state good.

However, current network administration and operation do not involve such kind of prediction and preparation. Generally the simulation on many combinations of topology and traffic model involves an immense amount of time and effort, hence is difficult, sometimes almost impossible. Current network operators utilize past history of the network state to roughly estimate and prepare for the near-future network state, in the hope that neither the new type of events occur, nor known events occur excessively. A typical example of the new events is a drastic change in the shape or the amount of network traffic of which the network have not experienced yet, such as caused by a policy change in the neighboring network. A typical example of the excessive known events is the occurrence of multiple link down events at the same time (link down events are classified in the known events in this discussion). Generally, both events are not prepared in today's communication network.

For the second objective, improving the routing systems, an environment to develop new routing systems and some means to evaluate routing systems are necessary.

One issue in the environment to develop new routing systems is that it is not easy to test a new routing algorithm. Most existing simulation tools employ the communication layer stack model, thus the new routing algorithm must sometimes be transformed to a distributed protocol entity, in order to be verified. Transforming an algorithm to a distributed protocol to fit in the tool's specific communication layer model is a bothersome task. Sometimes the global network topology is not available, or

many timers need to be prepared as a part of distributed protocol. This means that existing tools aim at verifying a protocol, and there is no tool for verifying an algorithm. We insist that it is a possible impediment in acceleration of routing technology research in communication network.

The evaluation of routing systems is a difficult task. This is because that it is not trivial to determine whether a routing system is good or bad, even when the routing system exhibits a good performance on a particular network condition. Routing systems may have preferences on network graph structures and network traffic models. For example, suppose that a routing system R_A runs better than a routing system R_B on a network graph structure G_1. It is not uncommon that on the other network graph structure G_2, the relation between the routing system R_A and R_B changes, i.e., R_B runs better than R_A. The same can be said for the shape of the traffic demand matrix, and for the setting of the routing metrics, instead of the network graph structure.

To alleviate this, with considerations to their preferences on network topologies and traffic models, many combination of topologies, routing metric settings, and traffic models must be tested and compared. In order to make it possible, many combinations of the network variables should be tested without excessive burden.

Routing research project in the past have verified their own technologies by either theoretical proof, by custom simulation and/or by existing simulation tools. We insist that all of them have problems. In verification by theoretical proof, usually only the core part of the algorithm is proven. In verification by custom simulation and by existing simulation tools, only limited network conditions are verified, because of the difficulty of existing simulation tools to simulate many network conditions extensively. In all methods, other aspects of the characteristics are left unverified, such as the compatibility to a specific kind of network topology.

The features required for the simulation tool to investigate many network states on various conditions, and to improve the routing system, are as follows.

- *Abstraction*: The support for the abstract perspective, such as done in Graph theory and Network-flow theory. Examples of this include the support for the traffic demand matrix and the network graph definition in an adjacency matrix.
- *Importation*: The ability to import network variables from various other tools. This improves the easiness to define an individual network variable, and in turn leads to the ability to handle a large-scale network, i.e., another aspect of the scalability of simulation tools, in contrast to the time taken to perform the simulation against the network size.
- *Independency*: Ability to define and maintain each network variables independently. This enables combinations of many instances of network variables, making the number of types of network conditions that can be produced for study huge.

A simulation tool that have these features may have the following additional features, which are the objective of this paper.

- *Productivity*: the ability to produce a wide-range of network conditions for simulation.
- *Development*: Easiness to construct a new routing algorithm in the simulation tool.
- *Comparison*: Ability to compare the consequences of routing systems in different network conditions. This contributes to the evaluation of routing systems. The purpose of *Comparison* includes the search of compatibilities between network variables and the routing system, hence *Productivity* is a prerequisite for the *Comparison*.

Next section studies other existing tools focusing whether the tool satisfies these features.

4. Existing Tools

We consider only the simulation tools in this section. The network design tools aim at a different objective, which is to design a network that satisfies specific conditions, such as the consequent latencies and the minimum investment against circuits' tariff. Also, the traffic engineering tools aim at a different objective, which is to determine a better network on a specific network parameters. Both of them neither satisfy the objectives of *Productivity* nor *Development*, because of lack of an important function, i.e., the capability to develop arbitrary algorithms and protocols. Hence, only the network simulation tools are the competitive candidate in this paper.

All the existing simulators in past target to simulate a close-to-real network state, such as available bandwidth and packet loss rate, in a

timed system. Their features are mainly focused on showing TCP performance and the consequences of detailed network protocol operations. Although they might be changed to achieve the objectives in this paper, their original purposes are different, and hence it is difficult to change them for the purpose of this paper. This leads to a motivation of developing a new routing simulation tool, SimRouting.

The summary of the comparison of simulation tools in required features is illustrated in **Table 4** later in Section 7.2, as a part of the evaluation of SimRouting.

4.1 Network Simulator NS-2

NS-2[19] is a famous network simulator targeted at network research. Various TCP algorithms, queues, moves of nodes, and radio propagation models are supported. NS-2 is written in C++, and employs OTcl scripts to describe the simulation scenario. *Abstraction* is not supported, because users must construct OTcl commands for each network links and for each traffic sessions. *Importation* in NS-2 is relatively manual in that users should write OTcl script, although users can construct arbitral graph construction algorithm by their own in OTcl. Sometimes graph generation tool supports NS-2, for example BRITE supports the output format to be included in NS-2. Although users can write separate OTcl scripts for each network variables, *Independency* is not supported in the sense that the definition of network traffic depends on the definition of network topology. Although BRITE supports the NS-2 format and an arbitrary script in OTcl is possible, *Productivity* is decided as not supported, because it is not the typical usage of NS-2, and still it is not easy to achieve *Productivity*. *Development* in NS-2 is hard because users must understand some abstract notions in NS-2, such as Agent and rtProto, and because OTcl interface must be provided for C++ program codes to be included in the simulation. *Comparison* is also not supported because of *Productivity*.

4.2 GloMoSim

GloMoSim[36] is a discrete-event simulation software targeted mainly for wireless network. It supports mobility models, radio models, and many Mobile Adhoc NETwork (MANET) routing protocols[16], in a layered fashion. It is based on a C-based discrete-event simulation language, named Parallel Simulation Environment for Complex Systems (PARSEC)[30]. Since GloMoSim targets to wireless network, GloMoSim handles the topology as spaces, rather than graphs. Hence, *Abstraction* is not supported. *Importation* from other tools is not supported. *Independency* is supported in the sense that users can write separate configuration files for each network variables. Although network topology (that is the node placement in this tool) can be automatically generated by inherent algorithms, *Productivity* is not supported, because of the simulation capability limited for wireless network. *Development* in GloMoSim is relatively easy since users must only write a single C module and hook `send`/`recv` functions on the global GloMoSim. However, due to the layered structure and space division approach of GloMoSim, accessing to the global topology seems not to be allowed. Thus, routing algorithms that are not defined as distributed protocol are not easily developed in GloMoSim. Hence, *Development* in GloMoSim is attributed to be hard. *Comparison* is not supported because of *Productivity*.

4.3 QualNet

QualNet[27] is a commercial derivative of GloMoSim. It is written in C++, and supports many advanced features such as wired network, real-time simulation, multi-threading, animation, and scalability of supporting 3500 nodes. QualNet has the ability to gather the network topology information from real networks via SNMP. QualNet enhances many feature to simulate or even emulate the real networks, and has the ability to cooperate with a battle simulator called OneSAF⋆1 Testbed Baseline (OTB)[23] and with a software for the Aerospace Industry called Satellite Tool Kit (STK)[1].

However, except for the SNMP support (*Importation*), evaluations on the required features remain the same with GloMoSim. *Abstraction* is not supported, since in QualNet users basically define network variables on a node or a link, and the network variables cannot expressed in a matrix. Many *Importation* method is supported, however for the purpose of this paper only the SNMP is achieved. *Independency* is not supported. *Productivity* is supported, since QualNet can import from many other tools, and can simulate even a mix of wired network and wireless network. *Development* is hard because of developers must study

⋆1 One Semi Automated Force (US Army Computer Generated Forces)

inside QualNet, and only testing of an algorithm seems to take time. *Comparison* is supported with the help of *Productivity* support.

4.4 OPNET Modeler

OPNET Modeler is developed in MIT, and released as a first commercial network simulator in 1987. Modeler supports most network protocols such as OSPF and MANET routing protocols. Modeler can import network models from vendor's network management tools, such as HP Software and CiscoWorks.

Evaluations of OPNET Modeler on the required features are similar to that of QualNet. *Abstraction* is not supported. *Importation* is supported, however for the purpose of this paper only the SNMP is achieved. *Independency* is not supported, since network variables are defined on top of the network topology. *Productivity* is supported, in the sense that importing from many NMSs can be performed. *Development* is hard because developers must study inside OPNET, and only testing of an algorithm seems to take time. *Comparison* is supported with the help of *Productivity* support.

4.5 OmNet++

OmNet++[22] is a discrete event simulation environment written in C++. It uses the newly defined language, called NED, to organize the simulation. *Abstraction* is not supported, and it utilizes the communication layer model. Although *Importation* in OmNet++ must be done manually either by text file or graphically, BRITE supports itself the exportation to OmNet++. *Independency* is not supported in OmNet++, because a model structure is described with dependency between the components in NED language. *Productivity* is not supported, as one must construct the model structure manually. *Development* is similar to GloMoSim, in that construction of new module is easy (write C++ module and register it by `Define_Module()` macro), but constructing an routing system without distributed protocol specification is difficult, due to lack of easy ways to access global topology. *Comparison* is not supported, due to the lack of *Productivity*.

4.6 Georgia Tech Network Simulator (GTNetS)

GTNetS[18] is a network simulation environment for researchers. GTNetS is a set of C++ objects, and the simulation scenario is written in the C++'s `main()` function. *Abstraction* is not supported, so the users must define topology manually in C++. *Importation* in GTNetS is basically manual, but it supports importing from BRITE. *Independency* is not supported, since the objects such as traffic definition must be defined on top of the nodes in the network. *Productivity* is not supported, because GTNetS supports only a few simple topology to be manually modified, other than BRITE. *Development* is easy in that users can write arbitral C++ program code with easy access to the global topology. *Comparison* is not supported due to the lack of *Productivity*.

5. Design and Implementation of SimRouting

This section describes the newly developed routing simulation tool, SimRouting.

5.1 Design Concept

The most important design concept in SimRouting is to achieve the easiness to manage simulations (called *Easiness* hereafter). The most simple motivation for this tool is to obtain the coarse-grained summary state of the network. We want to know what if the traffic changes, what if the link goes down, what if the routing metric is modified, and what if the routing algorithms change. In order to obtain the summary quickly, the most important property for a simulation tool is the ability to manage simulations very easily.

Second, another aspects of scalability for the simulator must be improved. If the user can define and perform the simulation of a moderate sized network without excessive burden, it indicates another aspect of scalability in the simulation. Capability to perform the simulation of a network quickly and easily means the tolerant to the increase of size of the netwrok.

Third, the tool must be simple. In order to support quick simulations, development of routing algorithms, and its evaluation, the tool's simplicity is important. Hence only a few, very simple models must be embeded, in order not to complicate the structure of the tool, to avoid the difficulity in simulations, development, and evaluation. This simplicity contributes to the desired property of the tool, that components are replacable, and hence is extensible.

To achieve these, SimRouting is designed so that it supports *Abstraction*, *Importation* and *Independency* of network variables. As mentioned earlier, these three properties enable the desired features for the simulation tool, i.e., *Productivity*, *Development*, and *Comparison*.

The precision of the simulator itself obeys the

`double` precision of IEEE 754[15]. With the hope of having fine precision in simulations, SimRouting uses C's `double` for its values in network variables. Actual precision of the result depends on the individual simulation.

SimRouting is a discrete-time simulation tool, which only simulates the instance of a time. This is in contrast to the descrete-event continuous time simulation employed in most existing tools. This means that the simplicity is prefered, rather than accuracy related to time, in the design concept of SimRouting.

5.2 Structure

The notion of network variables in SimRouting is illustrated in **Fig. 1**. Each network variables are stored in the modules, such as Network Graph Definition, Routing Metric Definition, Routing Algorithm, Routes, Traffic Definition, Network State. They are handled separately, from the Command Line Interface module at the top of the figure, achieving the *Independency*.

The network can be defined as an adjacency matrix, and the network traffic and routing metrics can be defined by the N×N matrix. This indicates the support for *Abstraction*.

5.3 *Easiness*

As mentioned in Section 5.1, *Easiness* is the most important design for the simulation tool. To achieve easiness in management of simulations, SimRouting implements an interactive shell with completions and help strings. This section describes the shell.

The users instruct simulations and resulting network status check, through Command Line Interface at the top of Fig. 1. Also, SimRouting can read a scenario file as a line-oriented list of simulation commands, which is handled by the Command Line Interface module. **Figure 2** illustrates the SimRouting interactive shell. `<cr>` and `<?>` are typed characters. Command line help is invoked by typing the `?` key. **Figure 3** illustrates the invocation with scenario files as arguments. Command line option `-i` specifies the run in interactive mode after execution of scenarios.

All of these features collectively enable the users of SimRouting to reach quickly to the appropriate commands the user should instruct. This means the *Easiness* to simulate communication network, and hopefully in turn contribute to improve the communication network.

5.4 *Importation*

This section descibes the implementation of

```
% ./simroutingcr>
cr>
simrouting> <?>
 logout     logout
 routing    enter routing node
 weight     enter weight node
 write      write information
 traffic    enter traffic node
 group      group specify command
 network    enter network node
 enable     enable features
 save       save information
 disable    disable features
 load       load configuration file
 show       display information
 quit       quit
 exit       exit
 graph      enter graph node
simrouting>
```

Fig. 2 SimRouting interactive shell

```
% ./simrouting <scenario-file1> <scenario-file2> ...
>& <logfile>
        or
% ./simrouting -i <scenario-file1> <scenario-file2>
...
        ⋮
simrouting>  (can execute subsequent commands.)
```

Fig. 3 SimRouting command-line invocation

SimRouting regarding *Importation* of three network variables: network graph structure, routing metrics, and traffic demands.

(1) network graph structure
Definitions of network graph can be achieved by importing from other tools such as BRITE, Rocketfuel, and SNMP, as well as by manual definition of each nodes and links. **Figure 4** shows the scenario file to import the network graph structure into independent graph variable, 100, 200, 300, and 400, from BRITE, Rocketfuel maps file, Rocketfuel weights file, and OSPF's LSDB via SNMP, respectively.
SimRouting also allows importation of a network graph from a routing sub-graph that consists of the routes for a specified destination. This contributes to visualize and to evaluate the result of a routing system, by exporting the abstract graph to other tools such as GraphViz[10], and by calculating the connectivity of the abstract graph using the existing algorithms in Graph theory. The importa-

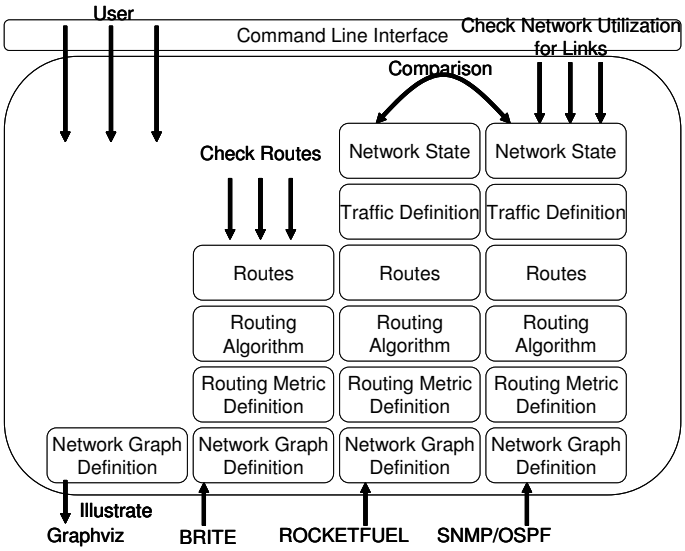


Fig. 1 SimRouting architecture

```
graph 100
 import brite etc/topology/RTBarabasi20UNI.brite
 show graph structure
exit

graph 200
 import rocketfuel maps ../rocketfuel/rocketfuel_maps_cch/1221.cch
 show graph structure
exit

graph 300
 import rocketfuel weights ../rocketfuel/weights-dist/1221/weights.intra
 show graph structure
exit

graph 400
 import ospf localhost public 0.0.0.0
 show graph structure
exit
```

Fig. 4 SimRouting graph importation

tion of the network graph from the routing sub-graph is used later in Section 6.2.

(2) routing metrics

Although a reference to the network graph is necessary to decide the value of N (i.e., the number of nodes) in $N \times N$ matrix, the routing metrics and traffic demands can be set and stored separately from the network graph definition. This means, basically, the network variables support *Independency*.

`Routing Metric Definition` (in Fig. 1) can be set by one of following ways.

(a) To set all link's routing metric as 1 (called *minimum-hop*).

(b) To set the routing metric for a link inversely proportional to the link's bandwidth (called *inverse-capacity*).

(c) To set all link's routing metric manually.

Depending on the method to define the routing metric, some definition of routing metrics may require the definition of a network graph in advance. In the case of inverse-capacity, capacities must also be included in the graph definition in advance, to define the routing metrics automatically by the inverse-capacity algorithm. Note that, for example, BRITE generates link capacities on generating a graph. The algorithm in the inverse-capacity method is similar to Cisco's recommendation[4], and calculates the metric c by the fomula $c = 100000/\text{bandwidth}$. Examples of definition of routing metrics (only the part regarding routing metrics in a scenario file) are shown in **Fig. 5**. The definition of routing metrics by inverse-capacity method does not achieve *Independency*, but achieves *Importation* instead, and contributes to achieving *Productivity*.

(3) traffic demands

Traffic demands can also be defined by calculation algorithms. This means the

```
weight 100
 weight-graph 100
 weight-setting inverse-capacity
 show weight
exit

weight 200
 weight-graph 200
 weight-setting minimum-hop
 show weight
exit

weight 300
 weight-graph 300
 weight-setting import rocketfuel ../rocketfuel/weights-dist/1221/weights.intra
 show weight
exit

weight 400
 weight-graph 400
 import ospf localhost public 0.0.0.0
 show weight
exit
```

Fig. 5 SimRouting routing metric specification

traffic demand variable also supports *Importation* from the algorithm.

An example of algorithm to automatically generate traffic demands is the model of Fortz and Thorup[11]. Note that Fortz-thorup model requires the specification of node location in the network graph in advance. This means that the use of Fortz-thorup model also negate *Independency* but achieves *Importation*, the same as in the discussion of routing metric definition.

Another ways to set traffic are to set a traffic demand in a random and specified pair of nodes. Setting a traffic demand in a random pair of nodes is called Single-random-flow. Users can utilize Single-random-flow multiple times to produce traffic demands easily. The use of Single-random-flow method is recognized as an calculation algorithm in this discussion, and hence is an example of *Importation*.

Random seed can be specified by "traffic-seed" command. Since it calls `srandom()`, setting random seed may help to reproduce the same traffic demands (both Fortz-thorup mode and Single-random-flow utilize random number generations).

Examples of traffic demand definitions are given in **Fig. 6**.

```
traffic 100
 traffic-graph 100
 traffic-seed 30
 traffic-model fortz-thorup alpha 100.0
 show traffic
exit

traffic 200
 traffic-graph 100
 traffic-seed 77
 traffic-set 1 2 bandwidth 40
 traffic-set random random bandwidth 60
 show traffic
exit
```

Fig. 6 SimRouting definition of traffic demands

5.5 *Development*

As an evidence of the feasibility to develop new routing systems and network related algorithms, this section introduces the systems developed on SimRouting.

Currently the following routing algorithms are developed in SimRouting.

(1) Dijkstra's shortest path calculation[8]
(2) Multipath route calculation of Deflection[35]
(3) Multipath routing algorithm called MARA[21]

Dijkstra is a classic algorithm, and Deflection and MARA are a new algorithms. SimRouting contributed to develop and to evaluate MARA.

Figure 7 shows examples of routing calculations. "show routing" command outputs the routes to standard output. Deflection calculates backup routes in a separate routing table, in addition to the same routes as Dijkstra. "show deflection set" command shows the contents of the separate routing table. "routing-algorithm ma-ordering" command specifies the use of MARA routing algorithm.

XXX, what reliability calculation is explained for here ?

The reliability calculation technology implemented in SimRouting is I_VT[29], one of the family of Sum of Disjoint Products (SDP) approach with Multiple Variable Inversion (MVI)[25]. Although the methods of path enumeration and sorting of path list are important for the computation time, path enumeration is performed by simple depth-first-search (DFS) each time in the graph, and the sorting of path list is not performed. These just mean that further performance improvement is possible for the computation time presented below.

Accuracy and precision.

```
routing 100
 routing-graph 100
 routing-weight 100
 routing-algorithm dijkstra
 show routing
exit

routing 200
 routing-graph 100
 routing-weight 100
 routing-algorithm deflection
 show routing
 show deflection set
exit

routing 300
 routing-graph 100
 routing-algorithm ma-ordering
 show routing
exit
```

Fig. 7 SimRouting route calculations

6. Example Simulation

In this section, example usages of SimRouting are shown, in order to exhibit the feasibility of SimRouting for the desired features.

6.1 Network state calculation

Here, Dijkstra routing is calculated on a BRITE generated topology of 20 nodes. The link bandwidth capacities generated by BRITE are used in the simulation, to derive inverse-capacity metrics, and to calculate the utilization of links. The node positions generated by BRITE are also used in illustration by GraphViz, and in traffic generation in fortz-thorup model.

Figure 8 shows the SimRouting scenario file for the simulation. This calculates the Dijkstra's shortest path routing on the BRITE generated topology. The purpose of the simulation is to investigate the utilizations of the links after routing the traffic demands on the network. **Figure 9** illustrates the topology. **Figure 10** shows the traffic demands generated by the fortz-thorup model. In showing the traffic demands, the fractional parts are rounded.

Table 1 is the resulted network state. Note that a bidirectional link in BRITE is represented as two unidirectional links in SimRouting. `s` and `t` denotes the source and sink node of the edge (network link). `Load`, `BW`, `Util` denote traffic loads, bandwidth capacity, and utilization of the link, respectively. Note that the values are rounded to the three decimal places in the table, while the calculations of

```
graph 100
 import brite etc/topology/sample.brite
 export graphviz tmp/brite-dijkstra-invcap.dot
exit

weight 100
 weight-graph 100
 weight-setting inverse-capacity
exit

routing 100
 routing-graph 100
 routing-weight 100
 routing-algorithm dijkstra
exit

traffic 100
 traffic-graph 100
 traffic-seed 30
 traffic-model fortz-thorup alpha 100.0
 show traffic
exit

network 100
 network-graph 100
 network-routing 100
 network-traffic 100
 network-load traffic-flows
 show network
exit
```

Fig. 8 SimRouting example scenario

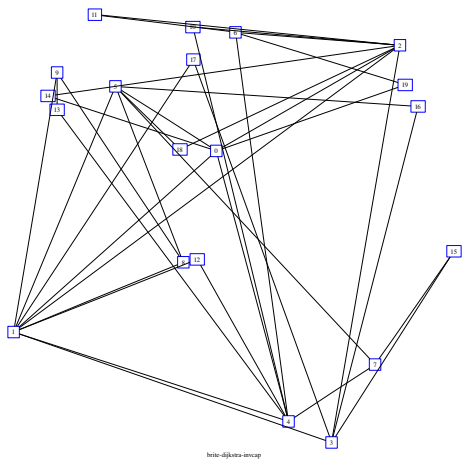


Fig. 9 SimRouting example network graph

such as maximum and minimum are not. The maximum link utilization in the network was 0.930688, and the minimum link utilization was 0.004501. The average and standard deviation were 0.239890 and 0.233147, respectively.

From Table 1, one can obtain, for example, how much the load of the maximally utilized link is, and which link it is. Executing successive simulations using SimRouting, one can

```
config-traffic-100>  show traffic
      0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19
 0   11   7   0   1   1   1   0   3   0   1  16  11   3   7   6   0   0   5   8  11
 1    7  11   0   1  13   5   6  10  26  10  15  18   2  21   3   4   0   2   5  32
 2   10  19   0   0  10   3   3   2   3   8   1  10   6  13  11   4   0   1   7  22
 3   15  11   0   2   4   1   2   2   5   3   2  17   2   9  12   2   0  10  15  24
 4   28  17   0   2   7  16   0   8  51   4  55  30   9  26  10  11   1  11   1   7
 5   28   2   0   0   8   7   4   5  12  14  21  49   2  15  13   3   0   8  43  12
 6   30  23   1   1  11   6   4  10  35   6  56  45   4  15  29   0   1   7  37  13
 7    6  31   0   1  15  13   9   3  14  10  27   1   5  12  10   1   0  11  48  25
 8   32  19   0   1   3   4   3   2  27  13  19  35   9  16   4   4   0   1  37  27
 9   17   5   0   0   8   4   2   5  27   1   7  40   5  11   6   5   0   7   1  11
10   18  14   0   1   4   9  12  11  20  10  44  13   7  16  10   4   1  11   3  35
11   25   8   0   1  13   3   3   9  30   5  10  12   7  14  17   8   1   6  49  17
12   42  19   0   2   0  23   0  20  35   2  58  10   5  24  17   1   1  14  68  27
13   22  21   0   1   4   3   4   3  13   2  28  11   4   9  19   3   0  10  24  21
14   31  12   0   0   2  10   3  12   5   2  28  55   0   7  16   3   0  16  44  29
15    1   4   0   1   5   0   1   2  10   4   4  10   1   2   7   1   0   1   6  12
16    6  12   0   0   1   4   5   5   7   4  14  12   0   8   2   5   0   4   7  14
17   12  17   0   1  14   4  13  10  18   4  43  48   0  16   1   2   1  19  59  14
18    4   3   0   0   0   6   1   4   5   3   6  11   3   5   1   2   0   5  10   2
19    1   0   0   0   3   5   1   3  10   0  16  19   1   5   3   1   0   4   2  15
```

Fig. 10 SimRouting example traffic demands

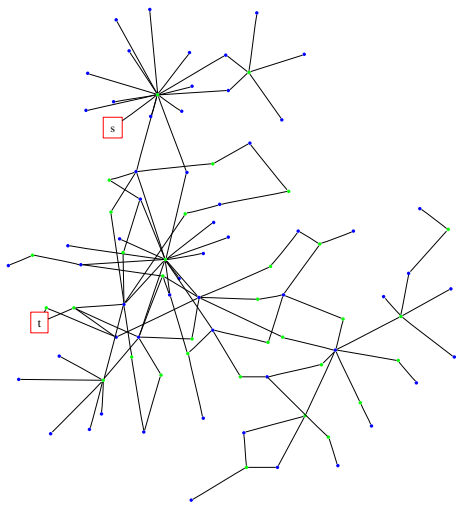


Fig. 11 *s-t* reliability calculation in the WIDE Project's network

further test how it is alleviated when a routing metric is changed, or how it becomes when traffic demands change.

6.2 *Comparison*

Calculation of a *s-t* reliability of a real network is given as an example in this section. The real network taken for reliability calculation is the WIDE Project's network[33]. It is illustrated by GraphViz in **Fig. 11**.

The source (`s`) and the sink (`t`) of the *s-t* reliability are the node 5 and the node 0, respectively. The scenario file used is shown in **Fig. 12**. "calculate reliability" commands in Fig. 12 instruct the reliability calculations. Calculating reliabilities were executed on the base graph, the dijkstra's routing sub-graph, and the MARA's routing sub-graph to see how the shortest path routing degrades the network reliability, and how the multipath routing preserves the system reliability of the base graph. All the link reliability is set to 0.9, which is a relatively low reliability than in practice, to emphasize the difference.

The statistics of the calculations and the reliability results are shown in **Table 2**. There are 3031 distinct paths from node 5 to node 0, where the minimum number of hops is 7, the maximum number of hops is 34, and the average is 20.632794. `#terms` column gives the number of terms finally involved in the mathematical formula, during the calculation of the reliability. The number of terms in calculating the reliability on Base Graph is as large as 208,843, and leads to the significant long computation time such as 4:37:29. Since there are large number of paths on Base Graph, the reliability is as high as approximately 0.864, despite the relatively low link reliability (i.e., 0.9). Notice that the Dijkstra's shortest path routing employed in the current Internet significantly decreases the reliability, despite the sufficient number of network paths in the base graph. In contrast to the reliability on the Dijkstra routing which is as low as 0.531, the reliability on the routing using a multipath routing algorithm called MARA increases significantly to 0.796. The reliability on the Routing Sub-graphs implies the reachability considering the faults that cannot be detected and avoided by the routing systems. The reliability indicates the reachabil-

Table 1 Results of example simulation in SimRouting

EdgeId	s	t	Load	BW	Util	EdgeId	s	t	Load	BW	Util
0	0	1	0.000	104.160	0.000	1	1	0	0.000	104.160	0.000
2	0	2	237.665	888.600	0.267	3	2	0	274.613	888.600	0.309
4	1	2	205.539	270.200	0.761	5	2	1	179.008	270.200	0.663
6	3	2	0.000	38.460	0.000	7	2	3	0.000	38.460	0.000
8	3	1	61.894	340.570	0.182	9	1	3	19.273	340.570	0.057
10	4	0	501.446	538.790	0.931	11	0	4	232.627	538.790	0.432
12	4	1	256.082	603.300	0.424	13	1	4	316.095	603.300	0.524
14	5	1	106.838	762.140	0.140	15	1	5	195.646	762.140	0.257
16	5	0	48.893	358.520	0.136	17	0	5	1.614	358.520	0.005
18	6	2	311.449	373.710	0.833	19	2	6	277.863	373.710	0.744
20	6	4	0.000	54.190	0.000	21	4	6	0.000	54.190	0.000
22	7	4	208.373	442.460	0.471	23	4	7	145.885	442.460	0.330
24	7	5	75.087	372.160	0.202	25	5	7	10.786	372.160	0.029
26	8	1	185.276	704.520	0.263	27	1	8	279.203	704.520	0.396
28	8	5	48.881	414.340	0.118	29	5	8	58.683	414.340	0.142
30	9	1	0.000	230.020	0.000	31	1	9	0.000	230.020	0.000
32	9	8	44.699	398.000	0.112	33	8	9	50.003	398.000	0.126
34	10	4	124.623	662.020	0.188	35	4	10	309.358	662.020	0.467
36	10	2	72.375	288.270	0.251	37	2	10	116.415	288.270	0.404
38	11	6	19.351	444.850	0.044	39	6	11	64.147	444.850	0.144
40	11	2	203.272	662.090	0.307	41	2	11	379.854	662.090	0.574
42	12	4	202.114	625.570	0.323	43	4	12	51.641	625.570	0.083
44	12	1	162.272	415.470	0.391	45	1	12	19.586	415.470	0.047
46	13	4	296.558	757.580	0.391	47	4	13	283.428	757.580	0.374
48	13	9	52.602	401.490	0.131	49	9	13	117.135	401.490	0.292
50	14	2	259.766	816.070	0.318	51	2	14	180.910	816.070	0.222
52	14	0	0.000	28.150	0.000	53	0	14	0.000	28.150	0.000
54	15	7	53.833	583.250	0.092	55	7	15	44.618	583.250	0.076
56	15	3	23.640	855.850	0.028	57	3	15	28.271	855.850	0.033
58	16	5	191.373	596.140	0.321	59	5	16	14.732	596.140	0.025
60	16	3	19.491	845.420	0.023	61	3	16	92.853	845.420	0.110
62	17	3	0.000	114.480	0.000	63	3	17	0.000	114.480	0.000
64	17	1	278.302	426.920	0.652	65	1	17	133.030	426.920	0.312
66	18	5	61.409	539.600	0.114	67	5	18	455.852	539.600	0.845
68	18	2	196.119	907.040	0.216	69	2	18	205.295	907.040	0.226
70	19	0	13.709	227.650	0.060	71	0	19	114.568	227.650	0.503
72	19	6	60.210	957.910	0.063	73	6	19	239.892	957.910	0.250

Util: routing-100: max: 0.930688 min: 0.004501 med: 0.467595 avg: 0.239890 std: 0.233147

Table 2 Results of the s-t reliability calculation in SimRouting

Graph used	#Paths	#terms in I_VT	Time Taken (sec)	Reliability
Base Graph	3,031	208,843	16,649	0.8641323325
Routing Sub-graph (Dijkstra)	1	1	0.000149	0.5314410000
Routing Sub-graph (MARA)	6	8	0.004171	0.7963936209

ity without relying on the routing response to the failure.

7. Evaluation

7.1 *Productivity*

Table 3 summarizes the options for each network variables. Only Dijkstra and Deflection can consider the setting of routing metric, hence

$$3 \times (2 \times 2 + 1) \times 2 = 30 \quad (1)$$

combinations of network setting can be constructed with only a few tens of lines in the scenario file. This feature can contribute a network researcher to evaluate routing algorithm and/or network setting with comparison to other similar settings. Changing only one network variable between two setting presents the influence of the changed network variable to the network state.

Note that the number of network types above (i.e., 30) is the number of method to construct a network setting. The number of actual network settings that can be imported in SimRouting is infinite even when we limit the importation of network graph from BRITE.

7.2 Achievement of Required Features

Comparison of features of SimRouting with other simulation tools is given in Table 4. The required features are from *Importation* (short-

Table 3 Supported methods in SimRouting

Graph	Routing metric	Routing algorithm	Traffic model
BRITE	minimum-hop	Dijkstra	Fortz-thorup
Rocketfuel	inverse-capacity	Deflection	Single-random-flow
SNMP/OSPF		MARA	

Table 4 Comparison of features with other simulation tools

Feature \ Name	NS-2	GloMoSim	QualNet	OPNET	OmNet++	GTNetS	SimRouting
Abstraction							
Importation (BRITE)	Yes	No	No	No	Yes	Yes	Yes
Importation (Rocketfuel)	No	No	No	No	No	No	Yes
Importation (SNMP/OSPF)	No	No	Yes	Yes	No	No	Yes
Independency	Yes	Yes	Yes	Yes	No	No	Yes
Development	Hard	Hard	Hard	Hard	Hard	Easy	Easy
Time	Yes	Yes	Yes	Yes	Yes	Yes	No
Accuracy	Yes	Yes	Yes	Yes	Yes	Yes	No

ened as `Graph` in the table) to Traffic Demands & Loads in Table 4. The rationale behind the required features is explained in Section 3.

Only a few tools support importation of network graph from other tools (i.e., *Importation*). Note that, since OPNET can import real network status from some vendors' network management tools, its `Graph (SNMP/OSPF)` is attributed to yes (support). However, SimRouting supports the largest methods to import network graphs, which is the desired property of a simulation tool for network researchers. Support of handling of network variables independently (*Independency*) is also supported in SimRouting, as well as done in most other simulation tools. *Development* is easy in SimRouting due to the easy access to the global network graph structure. SimRouting implements a method to calculate system reliability, while others do not (shown in the row of Reliability Calc). The ability to handle Routing Sub-graph is unique to SimRouting. All tools in the comparison support Traffic Demands & Loads.

To perform fair comparison, two measures that are important but yet deficient in SimRouting have been added in Table 4. These are *Timed System* and *Detailed Protocol Impl.*

Considering time in simulations is necessary to comprehend the difference and dynamism of a system over time. In contrast to the fact that most simulation tools support timed simulation scenario, SimRouting does not support simulation of time. This deficiency leads to the difficulty to handle time related variables, such as hardware failure probability (reliability of a hardware changes over time such as widely known as the bath-tub curve). In SimRouting, only a snapshot of a time can be taken into consideration. Thus, the failure probabilities of all network components at the snapshot time must be calculated manually in advance to the simulation, to consider the dynamical property of the simulated system.

Detailed Protocol Impl means whether the detailed protocol implementation is included in the simulation. Most other simulation tools take the approach of layered communication structure, where many detailed implementations of network protocols can be involved. The advantage of this approach is that the state of the network becomes closer to that of reality, because control traffic of the underlying network protocols are also considered. In SimRouting such overheads of the underlying network protocols cannot be considered in simulation. SimRouting is intended to be as simple as possible, to provide test environment for abstract ideas in theory.

In summary, SimRouting provides the simple test environment for new abstract ideas in networks and routing. In contrast to most other simulation tools that take layered communication protocol stack approach for the purpose to simulate the real network closely, SimRouting takes the approach of abstract network such as done in Graph theory. The abstract graph approach helps to understand the overview of the network, rather than details, and therefore more appropriate in searching new technology that has pervasive influence over the entire network, such as routing algorithms. Support of all required features in SimRouting enables comprehensive study of a new routing algorithm.

8. Conclusion

In order to maintain and improve the Internet performance, thorough investigation must be done with various network settings. Partic-

```
graph 100
 import ospf localhost public 0.0.0.0
 link all reliability 0.9
 show path source 5 destination 0
 calculate reliability source 5 destination 0
stat
exit

weight 100
 weight-graph 100
 import ospf localhost public 0.0.0.0
exit

routing 200
 routing-graph 100
 routing-weight 100
 routing-algorithm dijkstra
exit

graph 200
 import routing 200 sink-tree destination 0
 link all reliability 0.9
 show path source 5 destination 0
 calculate reliability source 5 destination 0
stat
exit

routing 300
 routing-graph 100
 routing-algorithm ma-ordering
exit

graph 300
 import routing 300 sink-tree destination 0
 link all reliability 0.9
 show path source 5 destination 0
 calculate reliability source 5 destination 0
stat
exit
```

Fig. 12 SimRouting reliability calculation scenario

ularly, routing systems including the methodology to set routing metrics are really important in terms of congestion avoidance.

Testing a routing algorithm on different network topologies was a difficult task because it was difficult to utilize the network model that is generated by other tools. A method that enables the evaluation of routing systems by comparison with other network setting was necessary.

A routing simulation tool named SimRouting is developed to help investigation and evaluation of various network settings. It supports the many network settings, such as by importing the network graphs from BRITE, Rocketfuel and SNMP/OSPF. The routing metric settings can be chosen from minimum-hop or inverse-capacity. The routing algorithm options are Dijkstra, Deflection, and MARA. The traffic model supports Fortz-thorup model and Single-random-flow. All network settings can be changed manually. The SimRouting tool supports the execution of simulation on various network settings easily, and hence contributes to improve routing systems.

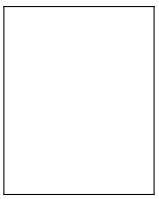
This work contributes to improve the Internet performance by providing a method to compare, evaluate and improve routing systems. The deficiency, preference and tendency of a routing system can be shown by SimRouting, and hence it can be utilized to improve routing systems. Improving routing systems certainly contributes to Internet performance.

References

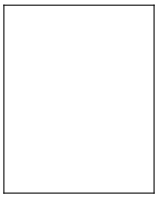
1) Analytical Graphics, Inc.: Analytical Graphics, Inc. (AGI), analysis software for land, sea, air, and space, http://www.agi.com/.
2) Beijnum, P. I. and Beijnum, I. V.: *Bgp*, O'Reilly & Associates, Inc., Sebastopol, CA, USA (2002).
3) Cariden: Cariden Technologies - IP/MPLS Planning and Traffic Engineering Software, http://www.cariden.com/.
4) Cisco Systems, Inc.: Cisco – OSPF Design Guide, <http://www.cisco.com/warp/public/104/1.html>.
5) Cisco Systems, Inc.: Cisco IOS NetFlow, http://www.cisco.com/en/US/products/ps6601/products_ios_pr
6) Cisco Systems, Inc.: Cisco MetroPlanner DWDM Operations Guide, Release 8.5, http://www.cisco.com/en/US/products/ps6095/products_configuration_guide_book09186a00808db813.html.
7) Cisco Systems, Inc.: Ciscoworks - Cisco - Cisco Systems, http://www.cisco.com/en/US/products/sw/cscowork/ps2425/index.html.
8) Dijkstra, E. W.: A note on two problems in connection with graphs, *Numerische Mathematik*, Vol.1, pp.269–271 (1959).
9) Doshi, B. T., Funka-Lea, C. A., Harshavardhana, P., Gong, J., Nagarajan, R., Ravikumar, S., Chen, S. and Wang, Y.: Integrated Network Design Tools (INDT): a suite of network design tools for current and next generation networking technologies, *ISCC '97: Proceedings of the 2nd IEEE Symposium on Computers and Communications (ISCC '97)*, Washington, DC, USA, IEEE Computer Society, p.332 (1997).
10) Ellson, J. and Gansner, E.: Graphviz, <http://www.graphviz.org/>.
11) Fortz, B. and Thorup, M.: Internet traffic engineering by optimizing OSPF weights, *IEEE INFOCOM*, Vol.2, pp.519–528 (2000).
12) Harrington, D., Presuhn, R. and Wijnen, B.:

An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, RFC 3411 (2002).
13) Hewlett-Packard Development Company, L.P.: HP Network Management Center - HP - BTO Software, https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119_4000_100__.
14) IBM: IBM Tivoli Software, http://www-306.ibm.com/software/tivoli/.
15) IEEE: IEEE 754: Standard for Binary Floating-Point Arithmetic, <http://grouper.ieee.org/groups/754/>.
16) IETF Working Group: Mobile Ad-hoc Networks (manet), http://www.ietf.org/html.charters/manet-charter.html.
17) j-sim.org: J-Sim Home Page, http://www.j-sim.org/.
18) Murray, B.: GTNetS - Home, <http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/index.html>.
19) NS-2 project: The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/.
20) NS-2 project: The Network Simulator - ns-2, http://www.opnet.com/solutions/network_rd/modeler.html.
21) Ohara, Y. and Imahori, S.: Computing a DAG using MA ordering for the all-to-one maximum flow routing problem, Mathematical Engineering Technical Report METR2007-09, Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo (2007).
22) OMNeT++ Community Site: OMNeT++ Community Site, http://www.omnetpp.org/.
23) OneSAF: OneSAF Public Site, http://www.onesaf.net/.
24) OPNET Technologies, Inc.: Network Planning — Network Engineering — Network Operations, http://www.opnet.com/solutions/network_planning_operations/.
25) Rai, S., Veeraraghavan, M. and Trivedi, K.S.: A Survey of Efficient Reliability Computation Using Disjoint Products Approach, *Networks*, Vol.25, No.3, pp.147–163 (1995).
26) RSoft Design Group: MetroWAND^{TM} Product Overview, http://www.rsoftdesign.com/products.php?sub=System+and+Network&itm=MetroWAND.
27) Scalable Network Technologies, Inc.: Scalable Network Technologies: Creators of QualNet Network Simulator Software, http://www.scalable-networks.com/.
28) sFlow.org.: Making the Network Visible, http://www.sflow.org/.
29) Tong, L. and Trivedi, K.S.: An improved algorithm for coherent-system reliability, *IEEE Transaction on Reliability*, Vol. 47, No. 1, pp. 73–78 (1998).
30) UCLA Parallel Computing Laboratory: UC LA Parsec Programming Language, http://pcl.cs.ucla.edu/projects/parsec/.
31) WANDL, Inc.: IP/MPLSView Home Page, http://www.wandl.com/html/mplsview/mplsview_new.php.
32) WANDL, Inc.: NPAT Home Page, http://www.wandl.com/html/npat/npat_% abnew.php.
33) WIDE Project: WIDE PROJECT, <http://www.wide.ad.jp/>.
34) Xiao, X., Hannan, A., Bailey, B. and Ni, L.M.: Traffic Engineering with MPLS in the Internet, *Network, IEEE*, Vol.14, No.2, pp.28–33 (2000).
35) Yang, X. and Wetherall, D.: Source selectable path diversity via routing deflections, *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, ACM Press, pp. 159–170 (2006).
36) Zeng, X., Bagrodia, R. and Gerla, M.: GloMoSim: a library for parallel simulation of large-scale wireless networks, *PADS '98: Proceedings of the twelfth workshop on Parallel and distributed simulation*, Washington, DC, USA, IEEE Computer Society, pp.154–161 (1998).

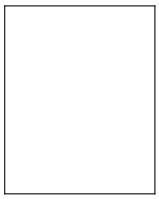
(Received ??? ??, ????)
(Accepted ??? ??, ????)



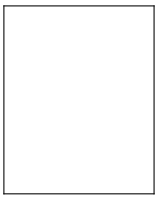
Yasuhiro Ohara received his Bachelor of Arts in Environmental Information in 1999, Master of Media and Governance in 2001, and Ph.D. in Media and Governance in 2008. He is now an assistant professor at Japan Advanced Institute of Science and Technology. He specializes in Internet Routing.



Masaki Minami received his B.A. and Master of Media and Governance from Keio University in 1996 and 1998. He is currently an Assistant Professor at the Graduate School of Media and Governance, Keio University since April 2005. His research interests include human communication on the Internet and its application with interdisciplinary approach.



Osamu Nakamura graduated from Keio University in 1982, Department of Mathematics, Faculty of Science and Technology, MS for Computer Science from Keio University in 1984, received Ph.D in Computer Science, Keio University, 1993. He is currently a Professor, Faculty of Environmental Information, Keio University since April 2006.



Jun Murai graduated from Keio University in 1979, Department of Mathematics, Faculty of Science and Technology, MS for Computer Science from Keio University in 1981, received Ph.D in Computer Science, Keio University, 1987. He is currently a Professor, Faculty of Environmental Information, Keio University since April 1997, and a Vice-President of Keio University since May 2005. His research interests include computer science, computer networks and computer communication.