Master's Thesis

Title

Proposal and evaluation of robust multipath routing with attractor seleciton

Supervisor

Professor Masayuki Murata

Author

Naotaka Onzuka

February 10th, 2014

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Proposal and evaluation of robust multipath routing with attractor seleciton

Naotaka Onzuka

Abstract

The rapid growth of information networks in size, complexity, and dynamics makes traditional and conventional mechanisms unsuccessful and unfeasible. As an example, a standard routing protocol called Open Shortest Path First (OSPF) [1] requires complete and consistent information about the whole network and computes shortest paths with Dijkstra algorithm. Whereas it can construct optimal paths under the given network condition, it incurs much control overhead in topology information exchanges and takes long time for path computation in a large scale network. Therefore, OSPF becomes infeasible for the large and dynamic network environment.

In our former research, we adopted a biologically-inspired nonlinear model, called attractor selection [2], to realize highly scalable, robust, and adaptive routing mechanisms [3, 4, 5]. It imitates adaptive gene expression of bacteria cells, which autonomously and adaptively synthesize nutrients in accordance with the environmental conditions. The proposal required lower computational cost and exhibited more robust and adaptive behavior than OSPF. However, because of a hop-by-hop based path construction, our proposal sometimes makes looping paths which require long time to resolve and lead to large delay and high loss probability. In this thesis, we take another approach to prepare multiple connected paths beforehand and adaptively choose the most suitable one from candidates, that is, multipath routing.

In multipath routing, each node pair has k-alternative paths and use them for the purpose of high robustness and load balancing [6, 7, 8]. In multipath routing, establishment of multiple paths and selection of a path used for packet transmission decide the robustness and performance of routing. From a viewpoint of robustness, paths should be as short as possible and as disjoint as possible. From a viewpoint of performance, a node should select a path leading to short end-to-end delay and high packet arrival ratio.

In this thesis, we consider three methods of multipath establishment and adopt the attractor selection model to path selection. We first compare multipath establishment methods in the robustness to link failures. Then, we compare our proposal with OSPF and mutlipath routing with greedy path selection from viewpoints of the average delay, packet arrival ratio, and path change ratio. Through simulation experiments, we show that attractor selection-based selection of path from a combination of link-disjoint and node-dispoint paths leads to the high robustness and performance.

Keywords

Multipath routing
Attractor selection
Robustness
Convergence time

Contents

List of Figures

List of Tables

1 Introduction

The rapid growth of information networks in size, complexity, and dynamics makes traditional and conventional mechanisms intended to achieve the optimal control and best performance unsuccessful and unfeasible. Optimization often requires the complete information about the targeted system. In a case of information networks, a central control unit or an administrator needs to collect the complete and up-to-date information about the whole network and conduct complicated computation to obtain the optimal solution. Even if control or decision is distributed among multiple entities, they require information consistent with each other to control and manage a network in a coherent way.

As an example, Open Shortest Path First (OSPF) [1], which is a standard Interior Gateway Protocol (IGP), derives shortest paths by using Dijkstra algorithm based on the topology information of a network. For this purpose, each node or each area exchanges a control message called Link State Advertisement (LSA), which contains information about adjacency relationship and link capacity, by means of periodic flooding. Flooding, where a message is repeatedly copied and forwarded to all neighbors by all nodes, is a powerful means of information dissemination. However, it takes time for LSA information to propagate to all nodes. Therefore, as the size of a network increases, the possibility of inconsistency in established paths becomes high. This fact further implies that OSPF cannot use dynamic link cost, e.g. delay and utilization, in derivation of shortest paths. Furthermore, OSPF also suffers from increased computational time. The computational complexity of Dijkstra algorithm ranges from $\mathrm{O}(E + N \log N)$ to $\mathrm{O}(N^2)$, where N and E are the number of nodes and links, respectively [9]. Because of these shortcomings, OSPF is inappropriate in a large and dynamic environment.

In our former research, we adopted a biologically-inspired nonlinear model, called attractor selection [2], and realize a highly scalable, robust, and adaptive routing mechanism [3, 4, 5]. The attractor selection model imitates adaptive gene expression of bacteria cells. Despite that the do not have any preprogrammed rule of adaptation, a cell autonomously and adaptively synthesizes nutrients to compensate nutrients missing in the environment and improve its growth rate. In [3], we applied the model to hop-by-hop routing for wired networks. We related selective nutrient synthesis to next-hop selection in packet forwarding. Each node calculates the attractor selection model and decides a next-hop node for a given destination node without the information about the whole network. As a consequence of autonomous decision of nodes, a packet is sent to a destination node. Assuming that the number of nodes in the network is N, the proposal required the computational cost in the order of N and exhibited more robust and adaptive behavior than OSPF. However, because of nature of hop-by-hop routing, our proposal sometimes suffers from looping paths. If path establishment fully depends on autonomous and independent decisions of nodes, even with limitation on the maximum number of hops, say h, the number of possible paths amounts to d^h where d is the average degree of node. In this thesis, we take another approach to prepare multiple connected paths beforehand and adaptively choose the most suitable one from candidates, that is, multipath routing. This approach however incurs two problems to solve: how to establish candidate paths and how to select a path to use.

In multipath routing, each node uses one of paths in candidates to transmit packets in accordance with network condition [6, 7, 8]. Here, both of establishment of candidate paths and selection of a path influence the robustness and performance. For example, paths established for a pair of nodes have common nodes or links, a single failure of node or link will disconnect all of paths and no path remains between them. Therefore, candidate paths should be independent from each other, that is, disjoint to prevent a single failure from affecting multiple candidate paths. Furthermore, when we assume that a link has the identical probability of failure, a longer path has the higher probability of being affected by link failure. The probability that a path consisting of l links is disconnected is estimated as $1 - (1 - p)^l$ where p is a link failure probability. Therefore, a nodes should as short candidates as possible to reduce a risk of disconnection.

Path selection can be regarded as an optimization problem. Existing proposals consider, for example, maximization of utility and maximization of max-min fairness [10, 11]. If a centralized control is possible, some heuristic algorithm can derive the optimal or sub-optimal solution. However, for the same reason that OSPF faces, such centralized optimization is not possible. Therefore, we need a method to accomplish the optimal or sub-optimal solution in a fully distributed manner. For this purpose, we adopt the attractor selection model which has the ability of distributed adaptation and optimization as explained above. By combining the multipath establishment method and the attractor selection-based path selection method, our proposal achieves both of robustness and performance.

The reminder of this thesis is organized as follows. First, we describe related work in section 2. In section 3, we describe attractor selection and introduce example of applications to routing.

In section 4, we propose multipath establishment methods and path selection method based on attractor selection. Then we conduct comparative evaluation these proposals in section 5. Finally, in section 6, we provide conclusion and future work.

2 Multipath routing

Basically, multipath routing is intended for higher reliability by holding an alternative backup path, better performance achieved by path diversity, and load balancing leading to congestion mitigation [10]. Current off-the-shelf routers such as products of Cisco and Juniper have capability of multipath routing, called ECMP. ECMP is an extension to OSPF. It splits traffic among paths which are equal in cost. In a case of path disconnection, traffic is concentrated on a remaining path. Therefore, it aims at load balancing and fault tolerance. However, paths are determined based on stable information and may not reflect the congestion state of the network.

Several multipath routing schemes have been proposed [6, 7, 8], which do not limit to equal cost paths. For example, in [6], authors propose Game Theoretic Stochastic Routing (GTSR) and show how many paths are needed for robust multipath routing in today's Internet. GTSR establishes multiple paths based on a stochastic flooding algorithm and decides traffic proportion distributed among the paths by resolving a maximum flow problem between each of source and destination pairs with the game theory. The authors show that GTSR can find paths by a simple algorithm and discuss tradeoff between robustness and packet loss rate. However, GTSR cannot reflect the network congestion, because a node decides the traffic proportion taking into account only traffic emitted by itself. On the other hand, in [8], authors propose a method to establish multiple paths based on global information and determine traffic proportion based on local information. They use an algorithm called Widest Disjoint Path (WDP). They define an indicator which reflects available link capacity of bottleneck links contained in a set of paths. WDP establishes paths which minimize the indicator by using global information about capacity of all links and traffic demand of all sessions in a network. To determine the traffic proportion among paths, they use an algorithm called Equalizing Blocking Probability (EBP). EBP equalizes the blocking probability of each path from observation of blocking probability of each path. Although a combination of WDP and EBP enables multipath routing adaptive dynamic changes in traffic demand, each node has to gather the global information frequently.

3 Attractor selection model

In this section, we briefly introduce explain the attractor selection model, on which our algorithm for adaptive selection of path depends. Then, we additional introduce some examples of application of the model to network control.

3.1 Mechanism of attractor selection

The attractor selection model is in the form of nonlinear stochastic differential equations. It is derived from autonomous and adaptive nutrient synthesis of *E.coli* cells in the dynamically changing environment of nutrient condition [2]. In [2], authors genetically modified an *E.coli* cell to have a metabolic network consisting of two mutually inhibitory sequences of chemical reactions corresponding to synthesis of two different nutrients. When one of the nutrients becomes scarce in the environment, the protein concentration activating a sequence for the missing nutrient eventually increases. Then, a cell stably maintains the state, i.e. gene expression, because it can compensate the missing nutrient and grow well. In [2], the model describing dynamics of protein concentrations m_1 and m_2 is formulated as,

$$\frac{dm_1}{dt} = \frac{s(\alpha)}{1+m_2^2} - d(\alpha)m_1 + \eta_1, \tag{1}$$

and

$$\frac{dm_2}{dt} = \frac{s(\alpha)}{1+m_1^2} - d(\alpha)m_2 + \eta_2, \tag{2}$$

$s(\alpha)$ and $d(\alpha)$ are functions of protein synthesis and degradation, respectively. Both of them depend on α which is called *activity* representing the cell activity or cell volume growth. Terms η_1 and η_2 are independent white noise in gene expression.

According to [2], equilibrium conditions in the metabolic network are called *attractors*. The above equations have two attractors, each of which corresponds to synthesis of one of nutrients. In a static environment a cell stays in a certain attractor, whereas it is influenced by internal and external noise. However, when the environment changes and affects the growth of a cell, it becomes unstable. Since instability allows a cell to stochastically change its gene expression, it eventually reaches a state, that is an attractor, which is suited for the new environment.

A generalized model can be formulated by Eq. (3). $\vec{m}$ is an M-dimensional vector and $\vec{m} = (m_1, ..., m_M)$, which expresses a state of a nonlinear system, e.g. bacteria. Each m_i is a scalar

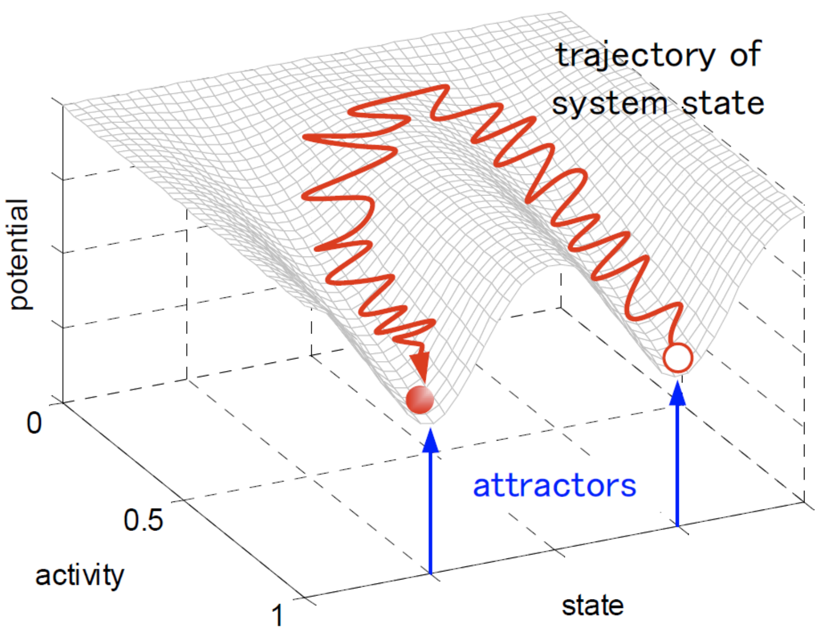


Figure 1: Behavior of attractor selection

and called state value. $f(\vec{m})$ is an energy function which defines the potential space and attractors. $\vec{\eta} = (\eta_1, ..., \eta_M)$ is a vector of noise elements.

$$\frac{d\vec{m}}{dt} = f(\vec{m}) \times \alpha + \vec{\eta} \tag{3}$$

Figure 1 shows a schematic image of the generalized attractor selection model. In this case, function $f(\vec{m})$ defines a double-well potential space. The x-axis is a projection of an M-dimensional state. Tails of arrows indicate states corresponding to attractors. The y-axis is the activity and the z-axis is the energy potential. The grid surface shows a potential space where a state of a system indicated by a ball moves along. A state always fluctuates slightly due to a noise term. However, when the activity is high, the dynamic is dominated by the term $f(\vec{m}) \times \alpha$ and a ball is trapped to one of attractors. When the activity becomes low on the other hand, due to for example a change in the environmental condition, the potential landscape becomes shallow and a ball moves more randomly than steadily. When it occasionally approaches an attractor suitable for the new condition, the activity slightly increases. The increased activity reinforces the potential landscape leading to further increase in the activity which plays a role of the driving force toward the attractor. Consequently, the new appropriate attractor is statically selected. As explained, the attractor selection model enables selection of a suitable attractor among possible stable conditions by switching stochastic behavior caused by noise term $\vec{\eta}$ and deterministic behavior controlled by potential function $f(\vec{m})$ with mediation of the activity.

3.2 Applications to routing

In this section, we introduce examples of applications of the attractor selection model, especially in the area of routing.

3.2.1 Mobile Ad Hoc Routing with Attractor Selection (MARAS)

MARAS [5] is an attractor selection based on-demand routing mechanism designed for Mobile Ad hoc Networks (MANET) to achieve high delivery efficiency. In MARAS, each node establishes and maintains a path to a destination following 2 steps.

1. Route establishment

A node establishes an initial path by disseminating Route Request Packet (RREQ) and Route Reply Packet (RREP) like Ad hoc On-Demand Distance Vector (AODV) [12]. When a demand to communicate with a certain destination comes up, a source node broadcasts a RREQ packet to neighbors. The RREQ packet is re-broadcasted by other node, that is , flooding. Each RREQ has a unique ID, which is used to detect and drop a duplicated RREQ. On relaying a RREP packet, each node remembers which node it comes from. When a RREQ arrives at a destination node, a RREP is generated. A RREP is transmitted over a reverse path to the source node by unicasting.

On the way of RREP to the source node, each intermediate node makes an entry for the destination node in its routing table. A routing entry has six fields shown in Table 1. A routing vector is in the form of a state vector of the attractor selection model. Initially, one state value corresponding to a neighbor from which it received the RREP is set at 1. All other state values are 0. In addition, the initial activity is set at 1 as well. When a source node receives a RREP, it first makes a routing entry. Then, it begins emission of data packets. At the source and intermediate nodes, a data packet is sent to a neighbor node having the maximum state value in a routing entry for the corresponding destination.

Table 1: Routing information of MARAS

Destination address	Address of destination node
Neighbor list $\vec{n} = (n_1, ..., n_M)$	Maintained by Hello message exchanges
Routing vector $\vec{m} = (m_1, ...m_M)$	State values of neighbors
Activity α	Goodness of current condition
Precursor list	List of address pairs of source and previous hop
Feedback window	Sliding window of hop count to destination

2. Route maintenance

MARAS maintains and updates a path as long as it is being used. In order to keep a routing table up-to-date, a source node has to know the latest condition of the current path. For this purpose, a destination node sends a feedback packet on reception of a data packet. A feedback packet traverses the reverse path of the corresponding data packet and informs a source node and intermediate nodes of the number of hops it took from the destination. A source node and intermediate nodes derive the new activity by using the following equation.

$$\alpha = \frac{\min_{t_0-T<t\leq t_0} w(t)}{w(t_0)}, \tag{4}$$

where $w(t)$ is the travelled hop count of the feedback packet received at time t. t_0 is the current time. Parameter T determines the duration that a node deposits past information. It is called feedback window and T is a feedback window size. Then, a node updates state values by using the following equation.

$$\frac{dm_i}{dt} = \frac{s(\alpha)}{1 + m_{max}^2 - m_i^2} - d(\alpha) \times m_i + (1 - \alpha) \times \eta_i, \tag{5}$$

where

$$\begin{aligned} m_{max} &= \max_{j=1,...M}(m_j), \\ s(\alpha) &= \alpha(\beta\alpha^\gamma + \varphi^*), \\ d(\alpha) &= \alpha, \\ \varphi^* &= 1/\sqrt{2}. \end{aligned}$$

η_i is independent white noise with mean of 0 and variance of 1. Parameters β and γ control the influence of activity over state values. The term $1 - \alpha$ additionally suppresses the effect of noise when the activity is high. If a path to a destination becomes long due to node movement for example, the hop count increases and the activity decreases. Then, the state values changes more randomly and as a result a next hop is selected randomly from neighbors to find a shorter path. When a shorter path is occasionally found, the activity increases and the diversity of state values are reinforced. As a result, a new and shorter path becomes stably selected.

Based on experiments simulating node movement and failures, it is shown that MARAS can achieve high delivery efficiency in a robust and adaptive manner. More details can be found in [5].

3.2.2 Attractor selection-based hop-by-hop routing for wired networks

In [3], we proposed a Hop-by-Hop routing mechanism for wired networks by adopting the attractor selection model to accomplish robust and lightweight routing in wired networks. It is based on MARAS, but we made improvements to fit to wired networks. First, since the length of a path in the number of hops does not reflect the status of a wired network such as congestion level, we defined the activity based on the end-to-end delay. It enables QoS-aware adaptive routing in a wired network, whereas OSPF employs static link cost. Second, we introduced a mechanism to reduce control overhead. Differently to MANET, where a path is established on demand, paths are required to always exist among all pairs of nodes in a wired network. Therefore, modification from a reactive mechanism to a proactive mechanism was required. In our proactive routing mechanism, each node periodically emits route control messages to all the other nodes for probing the quality of paths and updating routing tables. The amount of control messages considerably increases in a large-scale network and it must be reduced for higher scalability.

Figure 2 illustrates an overview of periodic exchanges of control messages and update of routing tables without control overhead reduction. Now assume that a certain node begins the procedure to update a routing table of a path to another node, say i. The former is a source and the latter is a destination in the following descriptions. The source first emits a route control message to the destination. The message takes the current path to the destination. On receiving the route control message, the destination sends back a feedback message to the source along the reverse path. A feedback message tells time when the route control message was received by the

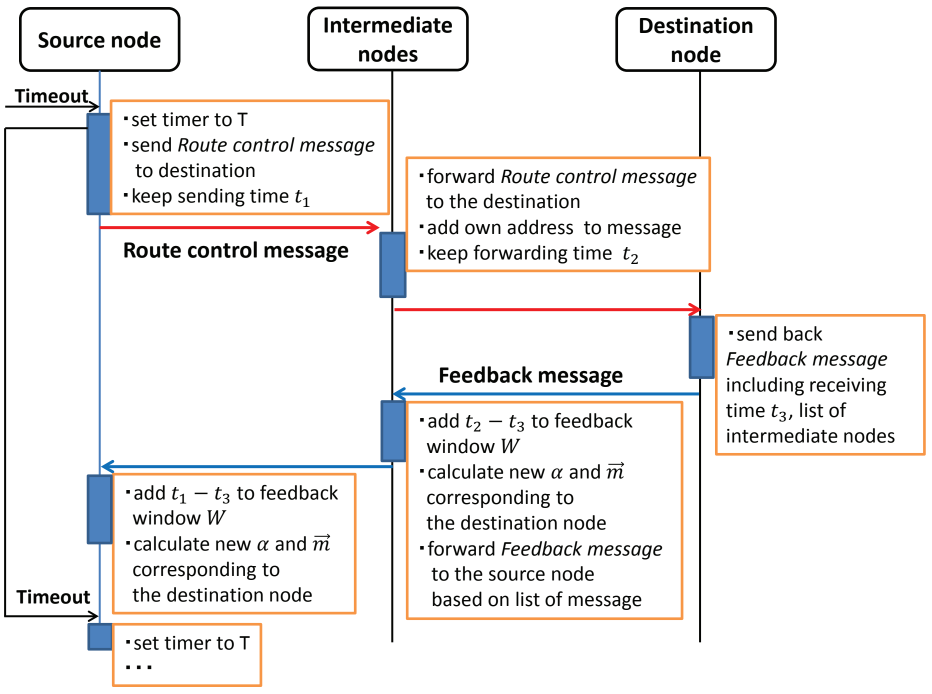

Source node
Intermediate nodes
Destination node
Timeout
・set timer to T
・send Route control message to destination
・keep sending time t_1
Route control message
・forward Route control message to the destination
・add own address to message
・keep forwarding time t_2
・send back Feedback message including receiving time t_3, list of intermediate nodes
Feedback message
・add $t_2 - t_3$ to feedback window W
・calculate new α and $\vec{m}$ corresponding to the destination node
・forward Feedback message to the source node based on list of message
・add $t_1 - t_3$ to feedback window W
・calculate new α and $\vec{m}$ corresponding to the destination node
Timeout
・set timer to T
・・・

Figure 2: Overview of periodic update routing information

destination. By using this information, nodes on the path can calculate the end-to-end delay to the destination. Next, the activity is calculated by using the following equation.

$$\alpha_i^{'}(h) = \frac{\min_{0 \le k \le W-1} d_i(h-k)}{d(h)}, \tag{6}$$

$$\alpha_i(h) = \begin{cases} \alpha_i^{'}(h), & \text{if } \alpha_i(h-1) \le \alpha_i^{'}(h) \\ c\alpha_i^{'}(h) + (1-c)\alpha_i(h-1), & \text{otherwise,} \end{cases} \tag{7}$$

where $\alpha_i(h)$ and $d_i(h)$ represents the activity and delay at h-th exchange of control messages. W is a feedback window size and c is a smoothing factor. Then, each node updates state values of the corresponding entry in a routing table.

$$m_{i,j}(h) = m_{i,j}(h-1) + \frac{s(\alpha_i(h))}{1 + m_{max}^2(h-1) - m_{i,j}^2(h-1)} - d(\alpha_i(h))m_{i,j}(h-1) + \eta_{i,j}, \tag{8}$$

where

$$\begin{aligned}
m_{max} &= \max_{j=1,\dots M}(m_j),\\
s(\alpha) &= \alpha(\beta\alpha^{\gamma} + \varphi^*),\\
d(\alpha) &= \alpha,\\
\varphi^* &= 1/\sqrt{2},
\end{aligned}$$

and $m_{i,j}(h)$ represents a state value corresponding to neighbor j for destination i at h-th exchange of control messages. $\eta_{i,j}$ is independent white noise with mean of 0 and variance of 1.

When the number of nodes is N, control messages are periodically exchanged among $N \times (N-1)$ pairs. To reduce the number of control messages, we proposed an overhead reduction method. As stated above, an intermediate node can update a routing entry for destination i by using a feedback message sent for the original source node, it does not need to emit a route control message for destination i by itself. Although details are not explained here, there are other occasions that a node can reuse a route control message and a feedback message exchanged among other nodes. By allowing such reuse, we could reduce the control overhead to 45%. Furthermore, we introduced a mechanism to mitigate a routing loop problem by passively and actively detecting a loop. It helped in increasing the packet delivery ratio from 56% to 98%. More details can be found in [3].

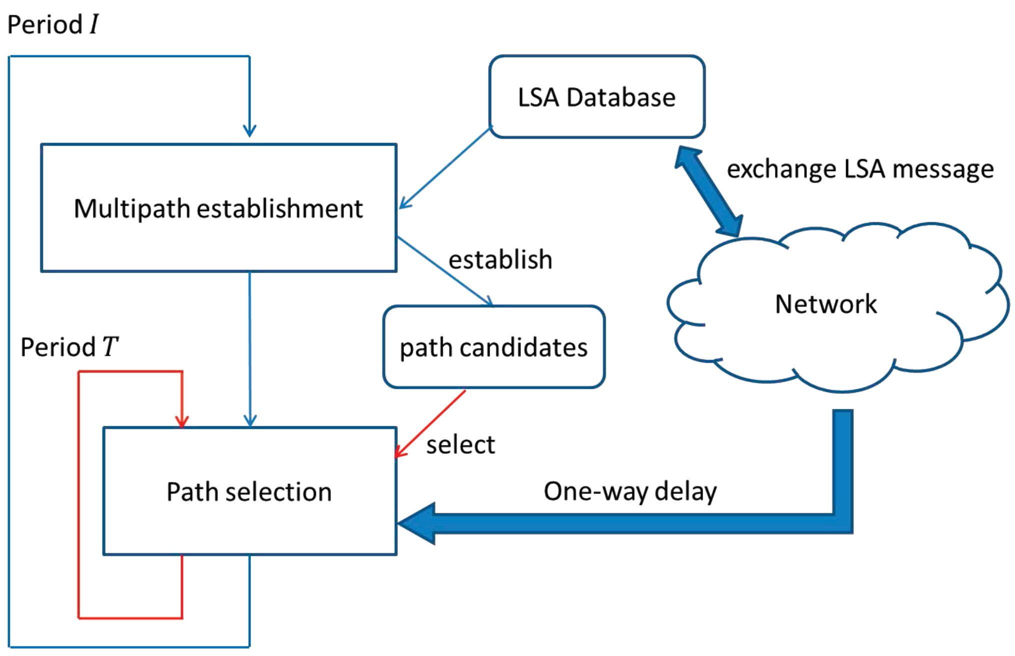


Figure 3: Overview of our proposal

4 Multipath routing with attractor selection

4.1 Outline of mechanism

Figure 3 illustrate an overview of our proposal. Our proposal mainly consists of two processes which are *Multipath establishment* and *Path selection*. The multipath establishment process derives candidate paths to all destinations based on network topology information at regular intervals of period I [s]. We should note here paths are not established in reality, which involves configuration of routing tables of all intermediate nodes. Since we consider so-called source routing, only a sender of a packet knows the complete path information, i.e. a set of links and nodes consisting a path. A sender embeds a list of nodes in a packet so that each intermediate node can know a neighbor to which it should forward the packet. The path selection process selects a path for packet forwarding from the created candidate paths at regular interval T [s]. On the contrary to the multipath establishment process which considers paths to all destinations, a node has path selection processes on a per-destination basis.

Before going into details of these processes, we first describe information that a node maintains for multipath routing.

Neighbor List

A neighbor list has addresses of neighbors, which a node is directly connected with. It is updated by Hello messages exchanged with neighbors with a period of T_{hello} [s]. When a node receives a Hello message from a node whose address is not in a neighbor list, it adds the sender's address to the neighbor list. Then, the node sets a *HelloTimeoutTimer* to $4 * T_{hello}$. When timer expires for not receiving any Hello message from the sender, the node removes the corresponding address from the list.

Destination list

A destination list maintains addresses of all nodes in a network for proactive path establishment and maintenance. A node recognizes other nodes through LSA message exchanges, which will be explained later. When a node finds a new node, it generates a new routing entry. When an LSA is removed, a node removes corresponding destination from the destination list.

Routing table

A routing table is a list of routing entries. An entry is created when a new node is added to a destination list and maintained for each of nodes in a destination list. An entry consists of a destination address, information about candidate paths, i.e. a list of nodes, an activity, state values, and a list of past measurements of the size W. When a node transmits a packet, a path with the maximum state value is selected among candidate paths established for the destination. An activity and state values are initially set at 0 and a random value ranging from 0 to 1, respectively.

LSA database

The multipath establishment process uses an LSA database to know the network topology. It is updated by reception of an LSA message. Each node periodically emits an LSA message containing an LSA of the node. An LSA includes an address of itself, a sequence number, the lifetime, and a neighbor list. The lifetime is initially I [s]. The lifetime decrease with passage of time. When the lifetime becomes to 0, a node deletes a corresponding LSA from an LSA database. Additionally, the node deletes corresponding destination from destination list and corresponding routing entry from routing table.

LSA sequence number

It is the an identifier of an LSA that a node creates. When a node creates a new LSA, it sets an LSA sequence number to the LSA for other nodes to detect duplicated reception of an LSA.

Topology change flag

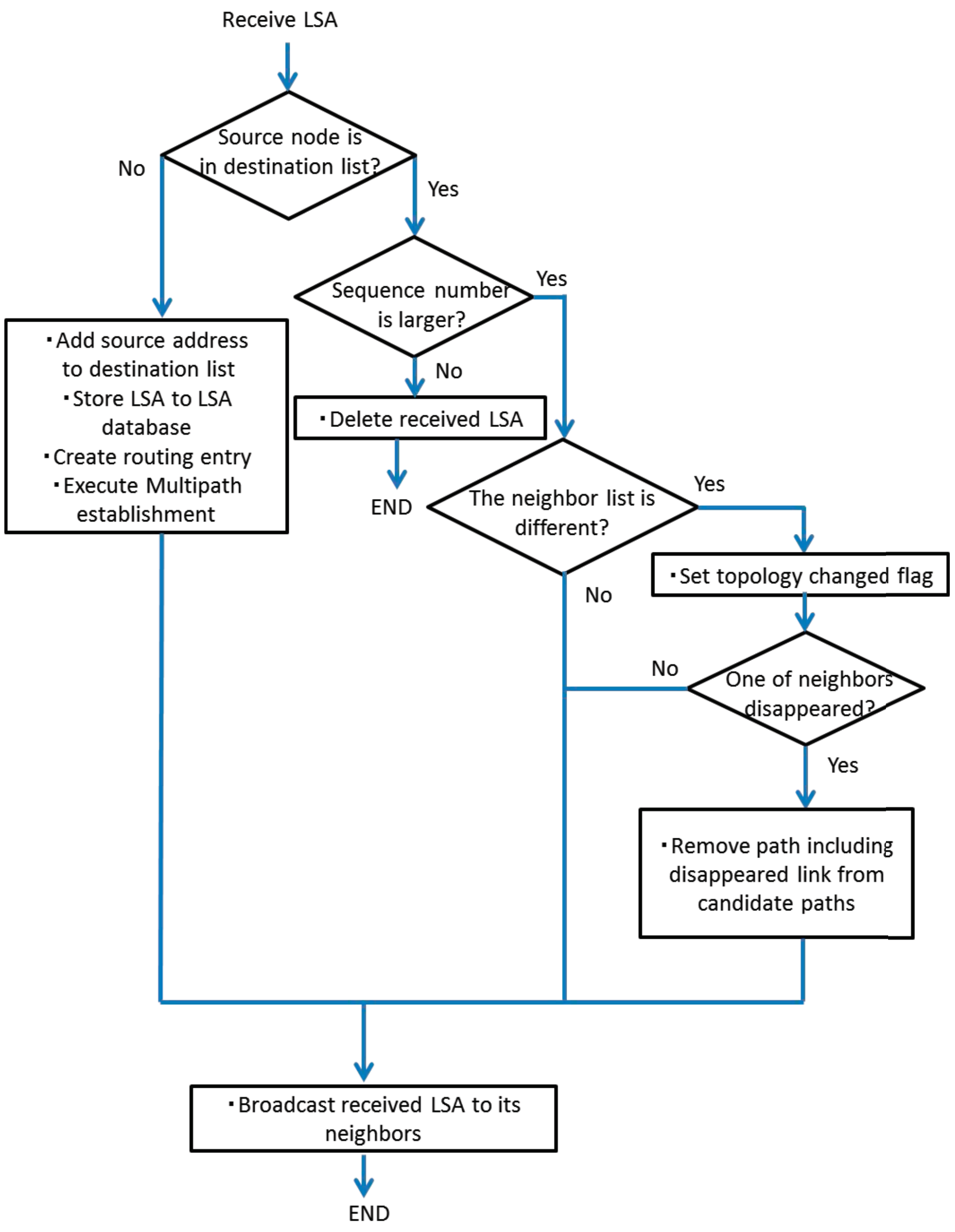
It is a boolean value to indicate whether topology change has occurred since the preceding multipath establishment.

Next, we explain a mechanism to maintain an LSA database, which is similar to that of OSPF. Each node generates an LSA at regular intervals of period I [s] or when a neighbor list changes. An LSA consists of an address of the node, a sequence number, the lifetime, and a neighbor list. Then, the LSA message including the LSA is sent to all neighbors. At the same time, the LSA is stored in a local LSA database. The process on receiving LSA message is shown Figure 4. When a node receives a LSA, the node checks whether the source address of the received LSA exists in the destination list or not. If the source address does not exist in the destination list, the node having received LSA adds the source address to destination list and adds the LSA to LSA database. In addition the node creates a routing entry to the destination and stores it in routing table, and then executes multipath establishment process explained in section 4.2. If the source address exists in the destination list, the node receiving LSA check whether the sequence number of received LSA is larger than that of LSA in LSA database. If the sequence number of received LSA is not larger than that of LSA in LSA database, the node delete received LSA and end this process. Otherwise, the node replaces the old LSA in the LSA database to received LSA. If the neighbor list of new LSA is different from old LSA, the node set the topology changed flag. If one of address of the neighbor list of old LSA does not exist in the neighbor list of new LSA, the node deletes paths including the disappeared link from candidate paths in all routing entry. After above processes, the node re-broadcast the received LSA to all neighbors.

Based on the above information, candidate paths are established and a primary path is selected. A multipath establishment process begins in the following cases.

(a) Periodic timer expires

A node has a periodic timer to regularly initiate the multipath establishment process. The timer is set at I [s]. When it becomes 0, it is reset to I again. It means that the timer



Receive LSA
Source node is in destination list?
No
Yes
Sequence number is larger?
Yes
No
・Add source address to destination list
・Store LSA to LSA database
・Create routing entry
・Execute Multipath establishment
・Delete received LSA
END
The neighbor list is different?
Yes
No
・Set topology changed flag
One of neighbors disappeared?
No
Yes
・Remove path including disappeared link from candidate paths
・Broadcast received LSA to its neighbors
END

Figure 4: Flow chart of LSA reception

precisely maintains the pace even if any of the following conditions is met. However, if a topology change flag is set on expiration of the timer, a node does not start the multipath establishment process.

(b) New destination is found

When a node finds a new node in a received LSA, it begins the multipath establishment process to make paths to the new node and update paths to other nodes expecting that traversing the new node results in a better path.

(c) All paths are lost for destination

When there is a path suffering from a link or node failure, it is removed from candidate paths. If no path remains for a certain destination, a node begins the multipath establishment process to rebuild candidate paths.

Therefore, in our proposal, a node does not establish, i.e. calculate, paths against topology changes caused by failures or others as far as it does not disconnect all candidate paths. On the other hand, all nodes need to calculate paths to all destinations even for a single and small change in the network topology.

A path selection process is executed always periodically with a period of T [s]. When a node detects a new node and establishes candidate paths, it does not begin a path selection process immediately. Since an LSA may contain multiple new nodes, we need to distribute timing of path selection to avoid synchronous behavior leading to instable control. For this purpose, a node defers execution for a random period ranging from 0 to T [s]. After a path selection process, a node execute a next path selection process in T_{rand}, where T_{rand} is random value ranging from $0.9 \times T$ to $1.1 \times T$. A node calculates T_{rand} every after a path selection process, to avoid synchronization of path selection.

4.2 Multipath establishment

As the way of multipath establishment decide the robustness of multipath routing mechanism as mentioned in section 1, we need to contemplate how to establish multipath to enhance robustness of our multipath routing mechanism. The GTSR [6] and WDP [8] could be suitable for this purpose, but the design concept of these algorithm is different from the purpose of robustness.

Therefore, we adopt k-shortest path because we can easily extend it to achieve high robustness. However, it is our future work to find the method that can find more robust multipath.

The k-shortest path algorithm [13] is an algorithm to find the k-th shortest path in a given graph. We extend k-shortest path to be more robust by following 3 methods.

(a) Node Disjoint (ND)

In ND, no pair of paths contains the same intermediate node. Therefore, a failure of a single node does not affect multiple candidate paths established among a pair of nodes, which leads to the high robustness of multipath routing. However, depending on the topology of a network, it is not always possible for all node pairs can find n node-disjoint paths. In that case, a constraint must be relaxed to have the sufficient number of alternative paths. The detailed algorithm is give below.

Step 1: A node finds shortest path by Dijkstra from complete topology information ($TI(1)$) and sets n to 1. Here, $TI(*)$ is initially complete topology information.

Step 2: If $n == k$, a node finishes this algorithm.

Step 3: A node deletes nodes included in n-th path from $TI(n + 1)$.

Step 4: A node tends to find path by Dijkstra from $TI(n + 1)$. If a node finds a new path, $n = n + 1$ and go to step 2. Otherwise, go to step5.

Step 5: Restore $TI(n + 1)$ to original topology information and set p to 1.

Step 6: If $p > n$, a node finishes this algorithm. Otherwise, a node set l to 1 and set H to the number of intermediate nodes in p-th path.

Step 7: If $l == H$, $p = p + 1$ and goes to step 6.

Step 8: A node tends to find a path from $TI(p)$ under the each condition which l nodes in p-th path can be shared. Once a node find a new path and the path is not completely same as j-th path ($1 \leq j \leq n$), a node increments n, set $T(n)$ to $TI(p)$ and delete unshared nodes in p-th path from $T(n)$, then if $n == k$ finish this algorithm.

Step 9: If $n == k$, a node finishes this algorithm. Otherwise, $l = l + 1$ and goes to step 7.

(b) Link Disjoint (LD)

LD does not allow sharing of a link among two or more candidate paths established for a certain destination. In comparison with ND, the length of paths are likely to be shorter because ND must take a detour to find a node-disjoint path. However, a naive algorithm of LD sometimes fails in finding n paths as ND does. Therefore, in that case, we allow paths to share the minimum and necessary number of links. The detailed algorithm is shown below.

Step 1: A node finds shortest path by Dijkstra from complete topology information ($TI(1)$) and sets n to 1. Here, $TI(*)$ is initially complete topology information.

Step 2: If $n == k$, a node finishes this algorithm.

Step 3: A node deletes links included in n-th path from $TI(n+1)$.

Step 4: A node tends to find path by Dijkstra from $TI(n+1)$. If a node finds a new path, $n = n+1$ and go to step 2. Otherwise, go to step5.

Step 5: Restore $TI(n+1)$ to original topology information and set p to 1.

Step 6: If $p > n$, a node finishes this algorithm. Otherwise, a node set l to 1 and set H to the number of links included in p-th path.

Step 7: If $l == H$, $p = p+1$ and goes to step 6.

Step 8: A node tends to find a path from $TI(p)$ under all condition which l links in p-th path can be shared. Once a node find a new path and the path is not completely same as j-th path ($1 \leq j \leq n$), a node increments n, $T(n) = TI(p)$ and deletes unshared links in p-th path from $T(n)$, then if $n == k$ a node finish this algorithm.

Step 9: If $n == k$, a node finishes this algorithm. Otherwise, $l = l+1$ and go to step 7.

(c) Node Disjoint and Link Disjoint (NLD)

NLD is a combination of ND and LD. First, the optimal and shortest path is established. Next, $\lfloor k/2 \rfloor$ paths are added from ND paths in an ascending order of path length, but except for the first path which is identical to the shortest path. Then, $\lfloor k/2 \rfloor$ paths are added from LD paths, but except for the first path and those identical to k/2 ND paths.

After k-shortest path is found by each multipath establishment methods, each paths found by the methods are stored to corresponding routing entry as candidate paths.

4.3 Path selection

A node selects a path to use for packet transmission by using the attractor selection model. Our path selection process is based on a next-hop selection method proposed in [3].

At regular intervals of T [s], each node transmits a route control message for each of destinations and a destination sends back a feedback message in response to a route control message. Based on a feedback message, a node derives the one-way delay of the current primary path. When a node does not receive a feedback packet, it decreases the activity by subtracting D where $0 < D < 1$. Since a path from a node to a destination can be different from a path from an intermediate of the path to the same destination, we do not adopt the overhead reduction method. Based on the measurement, a node calculates the activity and updates state values by Eqs. (6), (7), and (8). In comparison to [3] where a state value expresses the goodness of a neighbor as a next-hop node for a destination, in our proposal, it indicates the goodness of a candidate path. Then, a generated packet is sent to a destination over a path with the maximum state value.

5 Evaluation

In this section, we show the performance of our proposal by comparative evaluation.

5.1 Targets for comparison

We adopt the following two routing mechanisms for this comparison.

OSPF

OSPF always selects the shortest path and does not have alternative paths. Therefore, it is apparent that our multipath routing outperforms OSPF from a viewpoint of the robustness. However, our proposal does not necessary select the shortest path among candidate paths, which would lead to the longer delay.

Multipath routing with path selection

In multipath routing, path selection is in the form of an optimization problem. Existing proposals aim at maximization of the utility, the max-min fairness, and so on [10, 11]. In this paper, for the sake of simplicity and as a representative of path selection algorithms, we consider a greedy approach. In multipath routing with greedy path selection, each node has the same set of candidate paths as our proposal. It periodically measures delays of all candidate paths at an interval of T [s] and selects a path with the minimum delay. We should note here that measurement overhead is taken into account in the following evaluation.

In the following, *AS* stands for our proposal and *Greedy* does multipath routing with greedy path selection.

5.2 Simulation setting

We generate network topology for evaluation by using the Waxman model [14]. The Waxman model builds a flat structure imitating a real network, where a pair of distant nodes is less likely to have a link. The probability that a pair of node u and v, whose Euclidean distance is $d(u, v)$, has a link is given by the following equation.

$$P(u, v) = \mu \exp(\frac{-d(u, v)}{\nu L}), \tag{9}$$

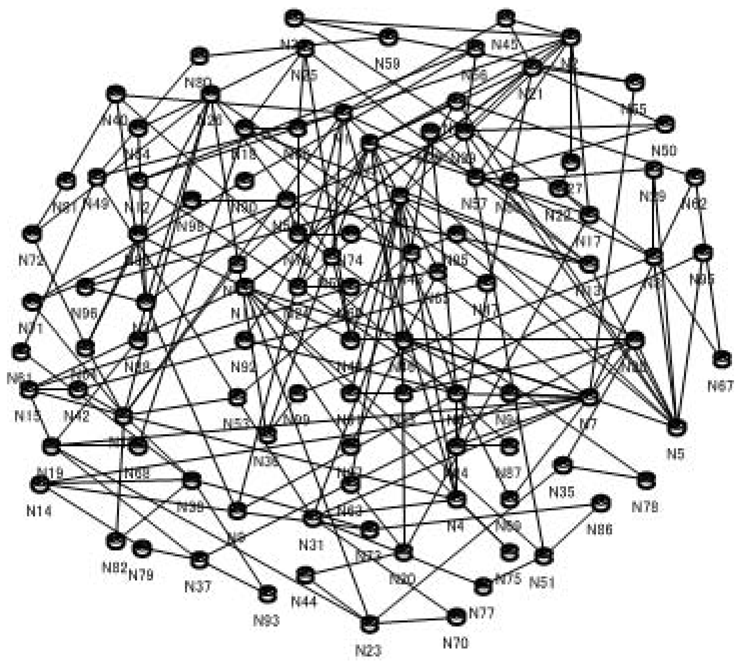


Figure 5: Example of Waxman topology

where L is the maximum distance between any arbitrary pair of nodes. μ and ν are parameters affecting the number of links where $0 < \mu$ and $\nu \leq 1$, An example of a generated network is shown in Fig. 5. Table 2 shows simulation parameters defining a network. Additionally we show the network topology setting we use in this evaluation in Table 2. We select pairs of nodes between which a CBR (Constant Bit Rate) over UDP (User Datagram Protocol) session exists. Although the most of traffic in a network employs TCP (Transmission Control Protocol) as a transport layer protocol, multiplexed TCP streams becomes similar to CBR in a router level network [15]. The data rate of each CBR traffic is set at 8 [Mbps] and packets are generated per 0.05 [s]. Node pairs are selected in a way that the total load on a network amounts from 0.52 to 0.57 in reference to [8]. The load is defined as $\frac{SHT}{LC}$ where S is the number of sessions, H is the average number of hops of the shortest paths of the sessions, T is the data rate, and L is the number of links in a network. We call a set of node pairs having traffic a traffic matrix.

We use Omnet++ version 4.1 [16] in our evaluation. As for OSPF, we use a code of INET framework with reference to RFC 2328 [17] and default parameters of a router of Cisco. We summarize parameters of our proposal in Table 3.

Table 2: Topology setting

μ	0.1
ν	0.58
Number of nodes	100
Number of links	219
Link capacity	50 [Mbps]
Propagation delay of link	100 [ms]
Packet buffer size	4.47 [Mbyte]
Packet loss rate per link	0

Table 3: Parameters of our proposal

Parameter	Value
I	10000 [s]
T	10 [s]
T_{hello}	10 [s]
k	3
W	20
β	10
γ	3
φ^*	$1/\sqrt{2}$
c	0.1
D	0.3

5.3 Performance measures

We use the following 4 measures for evaluation.

Disconnection ratio

We define a disconnection ratio at time t as the ratio of the number of node pairs whose candidate paths are all disconnected at time t to the number of all node pairs , i.e. 100×99 pairs. A disconnection probability indicates the robustness of routing, which is determined by path establishment.

Path change ratio

We define a path change ratio at time t as the ratio of the number of sessions which select another path than a path selected at time $t - T$ to the number of sessions. A path change ratio of 0 means that path selection converges and becomes stable at all sessions. A large path change ratio results in large delay jitter and packet disordering. In a case of a real-time communication application, a large play-out buffer is required to mitigate the influence of delay jitter and it spoils interactivity of two-way communication.

Average delay

We define an average delay at time t as an average of one-way delay of all packets generated at time t on all sessions. Since AS and Greedy tries to minimize the one-way delay, an average delay shows the goodness of path selection.

Packet arrival ratio

We define a packet arrival ratio at time t as the ratio of the number of packets which are generated at time t on all sessions and arrive at their destinations to the number of all packets generated at time t. Since delay indicates the level of congestion of a path, delay-based path selection should be able to avoid congestion and loss of packets. Therefore, a packet arrival ratio is another measure of path selection methods.

5.4 Robustness to failures

In this section, we compare three multipath establishment methods, i.e. LD, ND, and NLD, explained in section 4.2, and OSPF which has only one path between each node pair. In Fig. 6, the disconnection ratio is shown. The x-axis is the ratio of failed links to all links. We randomly

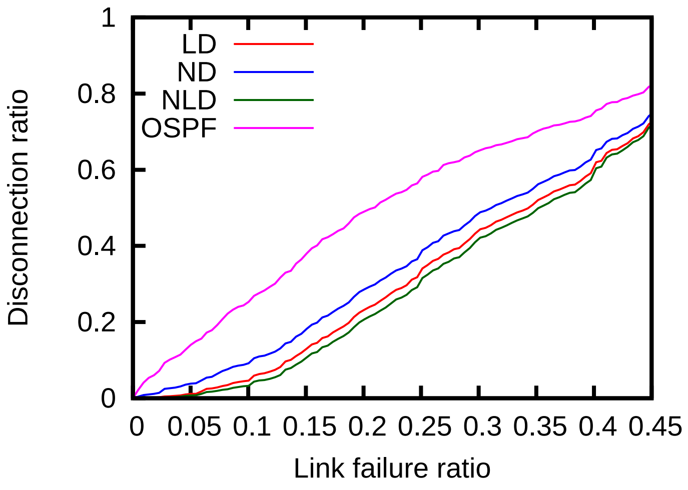
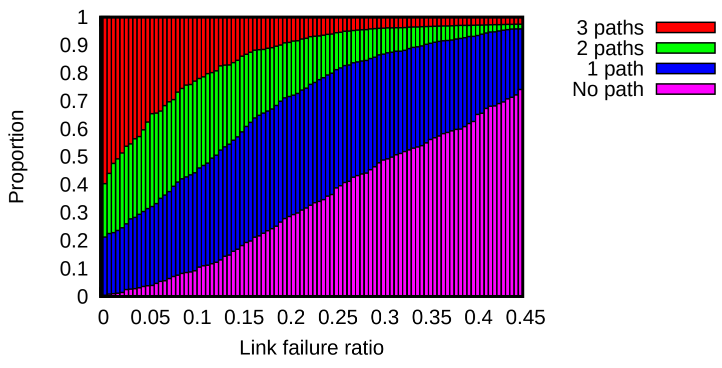


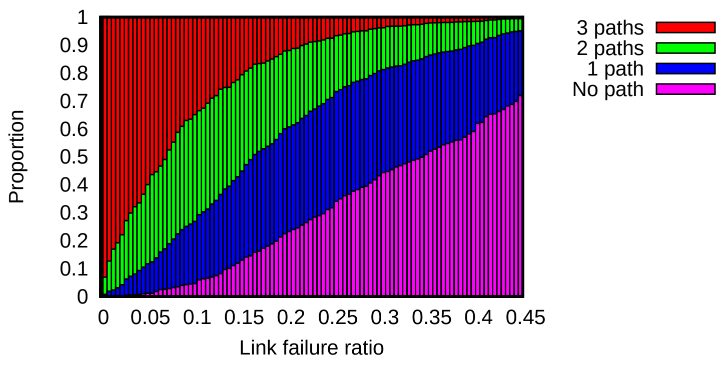
Figure 6: Robustness of link failures

removed links from a network one after another to simulate link failures, while keeping the connectivity of the whole network. Unsurprisingly, OSPF has the largest disconnection ratio among four. We notice that NLD has the smallest disconnection ratio. One of reasons is in a difference in the number of candidate paths that a node pair has. Figure 7 shows the number of candidate paths that each node pair has against the number of failed links. As shown, the proportion of node pairs which initially have three candidate paths is as much as 0.6 in ND, even when node sharing is allowed. This explains the larger disconnection ratio of ND than LD and NLD. In addition, Table 4 shows the number of node pairs whose two or all candidate paths share one or more links. Comparing LD and NLD, the number of node pairs whose two candidate paths share links is smaller with LD than NLD. However, the number of node pairs which do not have any disjoint candidate path is larger with LD than NLD (see 'among 3 paths'). It means that a single link failure can disconnect all candidate paths at once more often in LD than NLD. As a result, the disconnection ratio of NLD becomes slightly smaller than that of LD. In conclusion, to achieve high robustness, we need to establish as many disjoint candidate paths as possible.

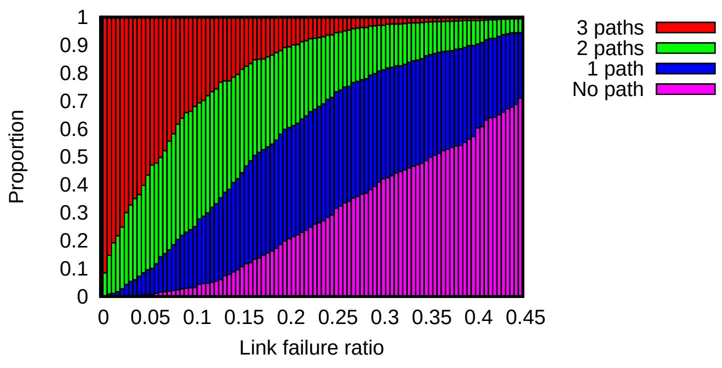
Next we compare path selection methods, i.e. AS and Greedy. Candidate paths are established by NLD in both cases. We evaluate temporal changes in the average delay, packet arrival ratio, and path change ratio when link failures occur. In Fig. 8, we show results where 5% of links are removed simultaneously at 600 [s]. To observe adaptive behavior of path selection, failed paths are selected so that there is no disconnected session after 600 [s].



(a) ND



(b) LD



(c) NLD

Figure 7: Proportion of number of remaining paths

Table 4: Link sharing among candidate paths

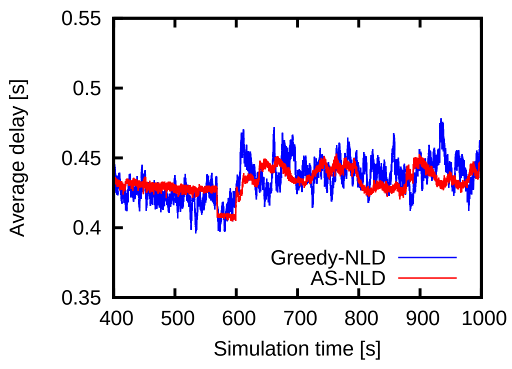
	ND	LD	NLD
among 2 paths	1476	1512	2054
among 3 paths	2370	2044	1932

In Fig. 8(a), it is shown that the average delay is similar among AS and Greedy. However, the average delay fluctuates more in Greedy than AS. That is, AS accomplishes more stable communication. Figure 8(b) shows that AS achieves the higher packet arrival ratio than Greedy at the beginning. When failures occur, the packet arrival ratio drops in both of AS and Greedy. The speed of adaptation and recovery is faster with Greedy than AS, but we can see the tendency that AS reaches the higher ratio than Greedy. Finally in Fig. 8(c), there is a significant difference between AS and Greedy. The path change ratio is much larger with Greedy than AS. In Greedy, most sessions change a path to use for packet transmission from one to another at every T [s]. When a node finds a path, say path A, other than the current one, say path B, has a shorter delay, it moves to path A. However, as the load on path A increases, the delay of path A increases as well. On the contrary, due to the reduced load, the delay of path B decreases. Therefore, at the next selection, path B seems more attractive than path A. Then, the node returns to path B. Under the competitive environment where sessions share links, sessions influence each other in path selection and it causes flapping. In conclusion, despite faster adaptation in Fig. 8(b), AS superiors to Greedy from viewpoints of stability.

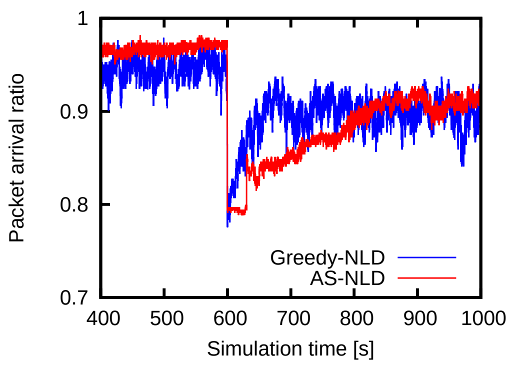
5.5 Performance in stable condition

In this section, we evaluate the performance of six combinations of multipath establishment and path selection. Simulation time 0 [s] is an instant when CBR sessions start transmitting UDP packets. We do not change a traffic matrix from the beginning to the end of a simulation run.

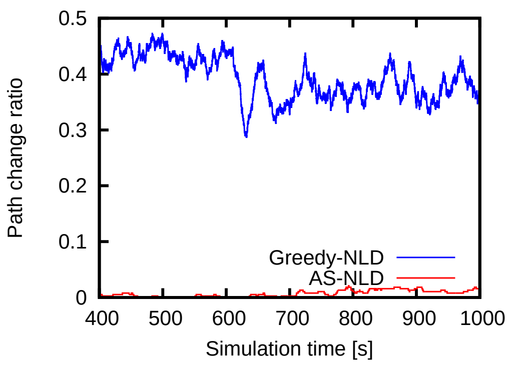
First in Figs. 9(a) and 9(b), path change ratios are shown for combinations of multipath establishment methods with AS and combinations with Greedy, respectively. Although paths often change at the beginning to find the best paths in AS, path change ratios gradually decreases for convergence of path selection. On the contrary, Greedy always suffers from flapping of path. Results of these differences can be found in Figs. 9(c) and 9(d), where average delays are shown. At the beginning, when AS searches for the best paths, the average delay is large independently of multipath establishment methods. However, as paths converge, delay becomes small and stable. On the contrary again, due to the flapping of path, the average delay fluctuates in Greedy. Figures 9(e) and 9(f) also support the superiority of AS to Greedy in terms of the packet arrival ratio. Around 500 [s], all of AS-LD, AS-ND, and AS-NLD achieve the similar level of packet arrival ratio.



(a) Average delay

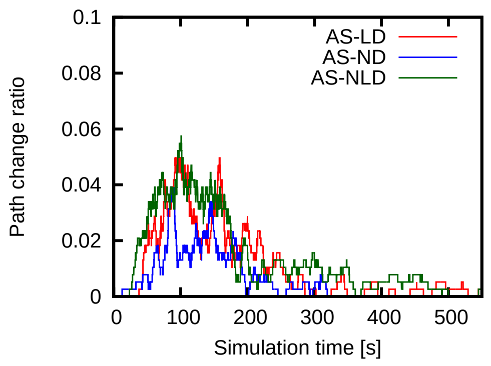


(b) Packet arrival ratio

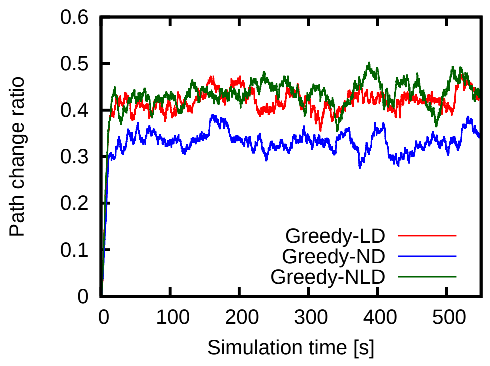


(c) Path change raito

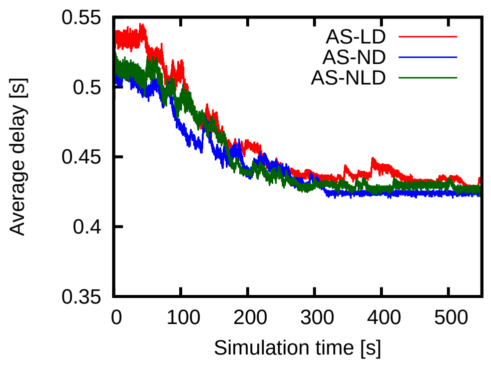
Figure 8: Comparison of path selection methods



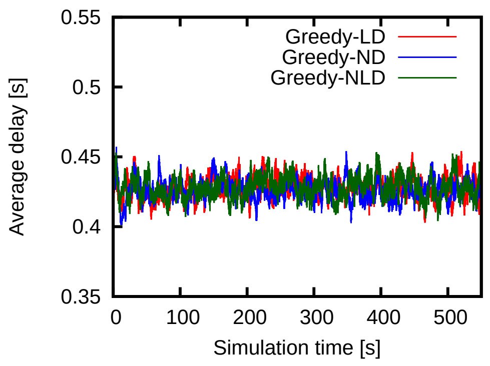
(a) Path change ratio of AS



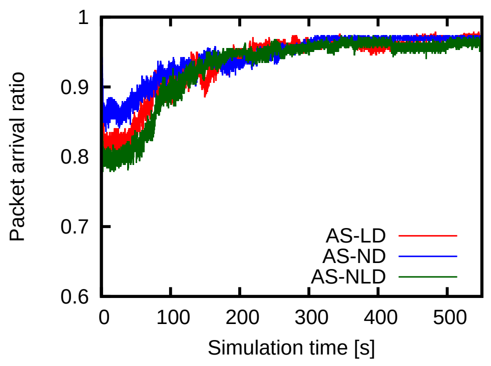
(b) Path change ratio of Greedy



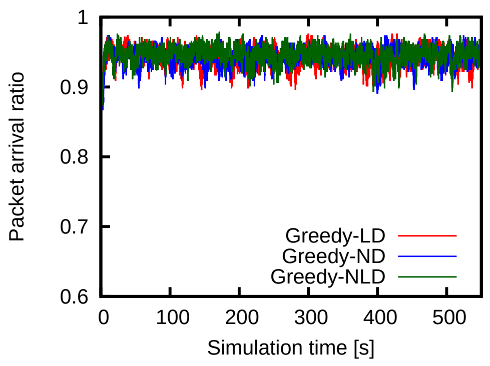
(c) Average delay of AS



(d) Average delay of Greedy

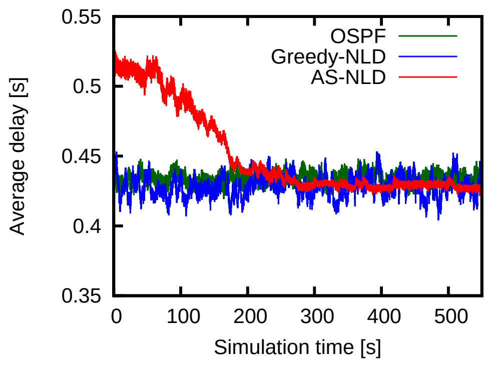


(e) Packet arrival ratio of AS

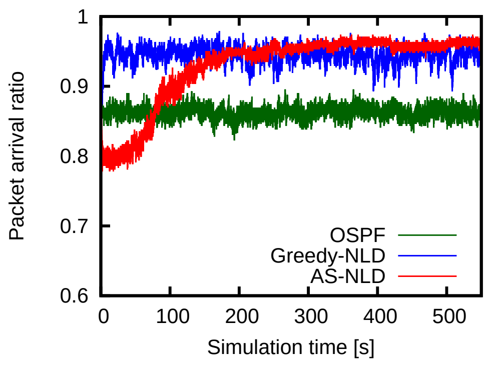


(f) Packet arrival ratio of Greedy

Figure 9: Performance in stable condition



(a) Average delay



(b) Packet arrival ratio

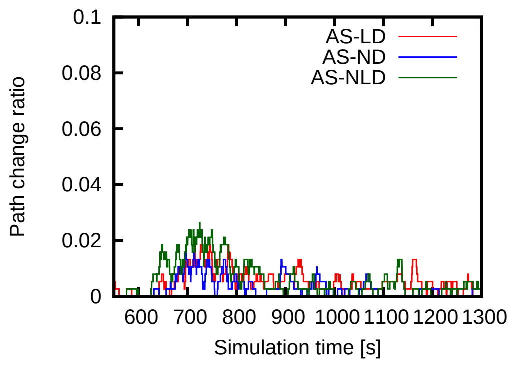
Figure 10: Performance comparison in stable condition

Finally in Fig. 10, we compare AS-NLD and Greedy-NLD to OSPF. Except for the initial transient period, the average delay is almost the same among three as shown in Fig. 10(a). However, AS-NLD has more stable delay than the others. In addition, AS-NLD achieves the highest packet arrival ratio in Fig. 10(b). A reason why OSPF results in the lowest packet arrival ratio is that OSPF insists on the shortest path. A Waxman network has the higher betweeness centrality than a random network. It means that there are nodes which many shortest paths contain. Therefore, there is concentration of load on certain nodes and links. However, OSPF cannot avoid them and suffers from congestion.

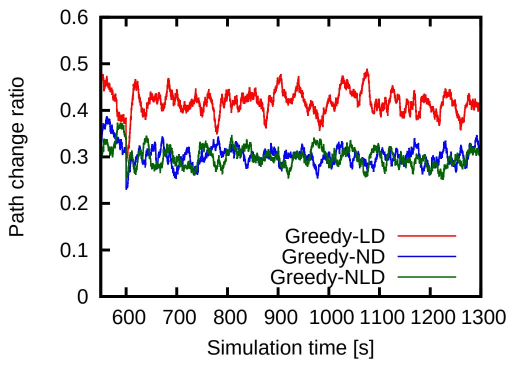
5.6 Adaptability to traffic changes

In this section, we change a traffic matrix and investigate adaptive behaviors. At 600 [s], i.e. the end of simulations in the previous section, we change a traffic matrix. Previous sessions are terminated and new sessions start.

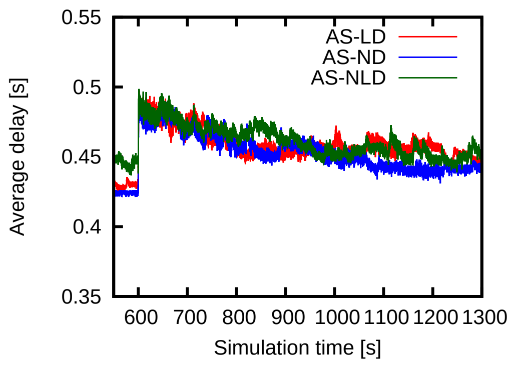
Figure 11(a) shows that path change ratios first increase in AS to adapt to a new traffic matrix. As time passes path change ratios gradually decreases, but it seems to require more time to reach convergence than Fig. 9(a). On the other hand, Greedy keeps flapping. In Fig. 11(c) the average delay of AS once increases. However as paths suitable for a new traffic matrix are selected, the average delay gradually decreases and becomes stable. On the contrary, the average delay of Greedy keeps fluctuating in Fig. 11(d). Figures 11(e) and 11(f) show the adaptive behavior of AS and more stable performance of AS than Greedy.



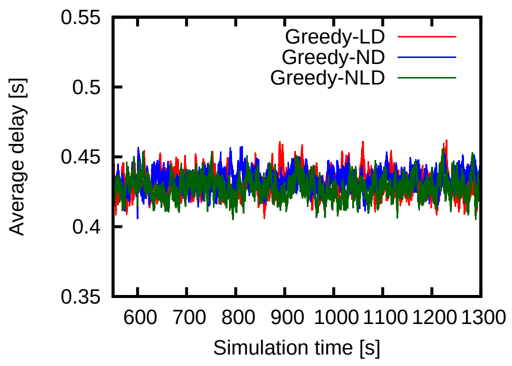
(a) Path change ratio of AS



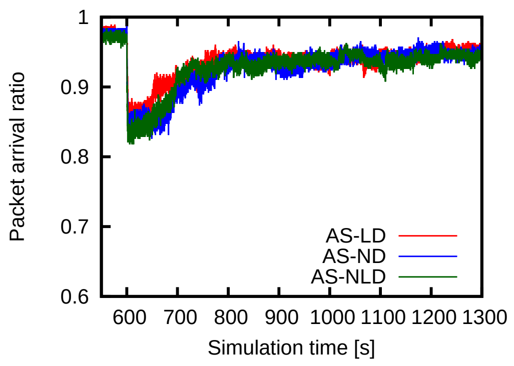
(b) Path change ratio of Greedy



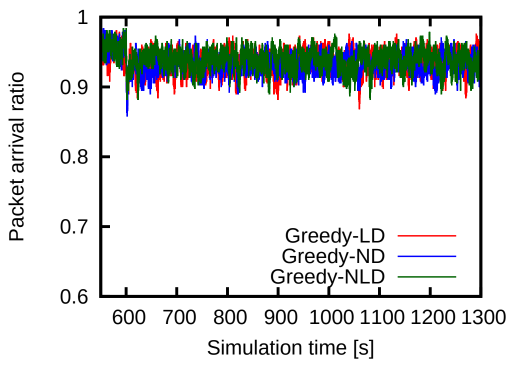
(c) Average delay of AS



(d) Average delay of Greedy

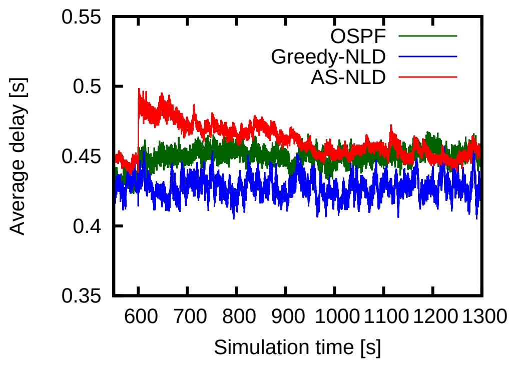


(e) Packet arrival ratio of AS

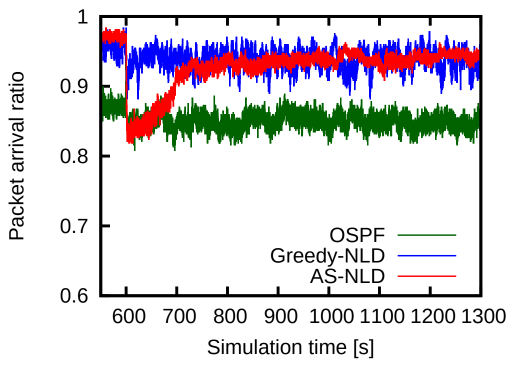


(f) Packet arrival ratio of Greedy

Figure 11: Performance in traffic change



(a) Average delay



(b) Packet arrival ratio

Figure 12: Comparison in traffic change

Finally in Fig. 12, we compare AS-NLD and Greedy-NLD to OSPF. As shown in Fig. 12(a), the average delay of Greedy-NLD is the smallest but it fluctuate the most. Figure 12(b) shows that the packet arrival ratio of AS-NLD is as high as that of Greedy-NLD.

In conclusion of the above results and discussion, while acceleration of convergence remains future work, AS is the most adaptive and stable method of path selection.

6 Conclusion

In this thesis, we propose the robust multipath routing mechanism based on attractor selection model. Simulation results show LD and NLD behave robust to link failure, and we indicate abundant paths and to suppress sharing are important for robust multipath routing. Furthermore, we show our proposal moderately adapts to the traffic change by very few change of path selection.

As future work, we search more robust multipath establish method by reference to the knowledge obtained in this thesis. Furthermore, we take additional comparative evaluation with other multipath routing mechanism, e.g. ECMP.

Acknowledgement

This thesis and my research life would not be accomplished without many people's supports. Firstly, I sincerely want to express my appreciation to my supervisor, Professor Masayuki Murata of Osaka University, for his guide to lead me to the place I'm standing. I also maximally appreciate Professor Naoki Wakamiya of Osaka University. Without his teaching, I cannot have grown as I am.

I would like to express my deepest appreciation to Associate Professors Shin'ichi Arakawa and Go Hasegawa, who teach me all kinds of educational things. I am also deeply grateful to Assistant professors Yuichi Ohsita, Yuki Koizumi, and Daichi Kominami of Osaka University, who gave me helpful comments. I want to express my gratitude to Mr. Takuya Iwai, Mr. Yuya Tarutani, and Mr. Yu Nakata. Without their supports, I could not achieve many things.

Finally, I thank my friends and colleagues in the Department of Information Networking, Graduate School of Information Science and Technology of Osaka University for their kindness.

References

[1] R. Coltun, D. Fergunson, and J. Moy, “OSPF for IPv6,” *RFC 2740*, Dec. 1999.

[2] A. Kashiwagi, I. Urabe, K. Kaneko, and T. Yomo, “Adaptive response of a gene network to environmental changes by fitness-induced attractor selection,” *PLoS ONE*, vol. 1, p. e49, Dec. 2006.

[3] N. Onzuka, N. Wakamiya, and M. Murata, “Robust and lightweight routing with attractor selection,” in *Proceedings of World Conference on Information Technology*, Dec. 2013.

[4] K. Leibnitz, N. Wakamiya, and M. Murata, “A bio-inspired robust routing protocol for mobile ad hoc networks,” *Computer Communications and Networks*, vol. 16, pp. 321–326, Aug. 2007.

[5] N. Asvarujanon, K. Leibnitz, N. Wakamiya, and M. Murata, “Robust and adaptive mobile ad hoc routing with attractor selection,” in *Proceedings of Adaptive and DependAble Mobile Ubiquitous Systems*, July 2010.

[6] C. Lim and S. Bohacek, “On the effectiveness of proactive path-diversity based routing for robustness to path failures,” in *Proceedings of Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, pp. 574–585, May 2008.

[7] P. Casas and F. Larroca, “Robust routing vs dynamic load-balancing a comprehensive study and new directions,” in *Proceedings of Design of Reliable Communication Networks*, pp. 123–130, Oct. 2009.

[8] S. Nelakuditi and Z.-L. Zhang, “On selection of paths for multipath routing,” in *Proceedings of International Symposium on Quality of Service*, pp. 171–184, June 2001.

[9] M. Sniedovich, “Dijkstra’s algorithm revisited:the dynamic programming connexion,” *Control and cybernetics*, vol. 35, pp. 599–620, 2006.

[10] H. Jiayue and J. Rexford, “Toward internet-wide multipath routing,” *IEEE Network*, vol. 22, pp. 16–21, Mar. 2008.

[11] V. Grout and N. Houlden, “Some notes and results on bandwidth-based routing and implicit load balancing,” in *Proceedings of 8th International Network Conference*, pp. 19–30, July 2010.

[12] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing,” *RFC 3561*, July 2003.

[13] Yen, Jin Y, “Finding the k shortest loopless paths in a network,” *Management Science*, vol. 17, pp. 712–716, July 1971.

[14] B. M. Waxman, “Routing of multipoint connections,” *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, Dec. 1988.

[15] J. Aracil and D. Morato, “Characterizing Internet load as a non-regular multiplex of TCP streams,” in *Proceedings of 9th International Conference on Computer Communications and Networks*, pp. 94–99, Oct. 2000.

[16] “Omnet++ network simulation framework.” `http://www.omnetpp.org/`.

[17] J. Moy and Ascend Communications, Inc., “OSPF Version 2,” *RFC 2328*, Dec. 1999.