A Heuristic Approach to Fair Routing Path Selection

Noriaki Hirosue, Kaori Yoshida, and Mario Köppen

Kyushu Institute of Technology, 680-4 Kawazu, Iizuka,
n_hirosue@pluto.ai.kyutech.ac.jp, kaori@ai.kyutech.ac.jp,
mkoeppen@ieee.org

Abstract. Knowledge about efficient usage of network resources is an important prerequisite for user collaboration in a static networked environment. Most of the network infrastructure internals cannot be directly sensed by users, except traffic allowances and response times. Here we will provide a technical approach for demonstrating to the user, how much changing traffic flows can guide to better employment of the shared network resources for all users together. The approach is based on order theory and fairness relations. Using the maxmin fair dominance relation, from its maximum set of maxmin fair states of random routings, the element with the largest throughput will be chosen, to yield a fair routing. The reported experiments will demonstrate that, for example, for a group of about 10 users, 500 samples are sufficient to report a suitable fair traffic allocation to the users.

Keywords: maxmin fairness, network routing, heuristic method, fair routing

1 Introduction

In the field of network design, consideration of fairness among collaborating users is of high importance. For network fairness, there are two possible points of view. The first is the network infrastructure point of view, such as traffic analysis, smart routing and network resource allocation. For example, Nakamura et. al. [7] proposed a *global network measurement platform*, which achieves a simultaneous and efficient use of resources. The proposed global network measurement enables integrated network monitoring functionality on-demand and adapts it to specific application purposes. Mo and Walr [6] demonstrated the existence of fair end-to-end window-based congestion control protocols for packet-switched networks with *first come-first served* routers. They studied congestion control protocols based on the concept of generalized proportional fairness. The infrastructure-based network fairness research has a long history [4] and it is still a hot topic. The second is the network user point of view. For example, an important factor here might be the experienced response time of a network game, QoS in general, or the sense of closeness in a social network. Verschae et al. [9] tried to improve QoS in network systems based on partial user-supplied information. Considering network fairness from user point of view means to evaluate user satisfaction. Besides, once we can infer user satisfaction or intentions, the network system can provide the best environment that individually suits to each user. At the same time, users will be able to choose their traffic flows and routings in a similar way like when people avoid traffic jams by using a car navigation system.

In this paper, we propose a criterion for fairness based on two-stage use of the *maxmin fair dominance relation*. Thus, we can provide an evaluation method for network fairness considering both, the infrastructure and the user point of view. The approach is based on comparing a number of maxmin fair states of user traffic for a number of randomly selected routings according to maxmin fairness, then deriving its maximum set, and finally selecting the maxmin fair state with maximum throughput from this maximum set.

In section 2, we will recall the necessary formal and technical background, before proposing this Throughput Maximizing Fair State (TMFS) selection approach in section 3. The approach is validated by experiments given in section 4. The paper concludes with a short summary and an outlook.

2 Technical Background

In the following, we will focus on transportation networks (data networks, flow networks, or communication networks). A network is represented as an undirected graph $G = (N, L)$ with node set N and link set L (allowing traffic flow in both directions of a link, and the graph is assumed to be connected). There is also a set P of m sender-receiver pairs $p_i = (s_i, r_i)$, and each sender s_i wants to send data flow (measured in amount per time unit, for example, the average number of packets per time unit) to its corresponding receiver r_i. Note that sender and receiver may overlap, so traffic can be send from different senders to the same receiver, or from the same sender to different receivers. For sending traffic through the network, a loop-free sequence of links from each sender to its receiver has to be specified. For connecting pair p_i by a sequence of links L_i, the destination node of a link is the source node of the next link, the source node of the first link is the sender s_i, and the destination node of the last link is the receiver r_i. Such a sequence of links is called a *path*. The total set of all paths for all $p_i \in P$ is called a *routing*. For simplification, we will also refer to a sender-receiver pair and their connecting path as a *user* of the network.

Furthermore we will assume that this user end-to-end traffic flow, also called *throughput*, which can be send via a link is limited by a maximum capacity c_{ij} for the link connecting nodes i and j. Thus, if for a specific routing several sender-receiver pairs share the same link, then the maximum capacity is limiting the *sum* of the traffics that can be send by all link-sharing senders together.

In order theory, for a given set A, a *binary relation* $>_r$ between elements of A is a subset of $A \times A$: a pair (x, y) belongs to this subset if and only if $x \in A$ is in this relation to $y \in A$ (i.e. $x >_r y$). Then, for each relation we can define the *maximum set* and the *best set* subsets of A (both are possibly empty). The maximum set contains all elements $x \in A$ such that there is no different

$y \in A$ with $y >_r x$. The best set contains all $x \in A$ such that for any other y $x >_r y$ holds. For the standard size relation of real numbers, for example, both definitions coincide, but for other relations, they will refer to different sets. In many cases, the best set of a relation is usually empty, while the maximum set usually contains more than one element.

A mapping from the set of all routings of all networks to subsets of all routings of all networks is called a *routing path selection*. If these subsets correspond to the maximum sets of a relation, following Suzumura and the case of social choice theory [8], the routing path selection will be called *rationalizable*. Thus, the task of routing path selection is two-fold: specify a rationale for the selection by means of specifying a relation, and provide a method for finding (or approximating) the maximum set of this relation among all routings as well.

2.1 Bottleneck flow control

To assign a traffic flow t_i to each sender-receiver pair p_i for a given routing, a suitable control paradigm for the network is needed (in the following, the totality of such traffic assignments for all users will be called a *state*). Usually this would be an optimization task, for example maximization of the total throughput of all sender-receiver pairs. But it is known for long that such a control paradigm can lead to the exclusion of users. As a simple example, consider a routing where one user u_1 shares one link with capacity c with user u_2, and one link with same capacity c with user u_3. If user u_1 sends the flow t by such a routing, users u_2 and u_3 can each send at most $c - t$, and the total traffic is $t + 2(c - t) = 2c - t$. This value is maximal for $t = 0$, i.e. allowing no traffic for user u_1 at all.

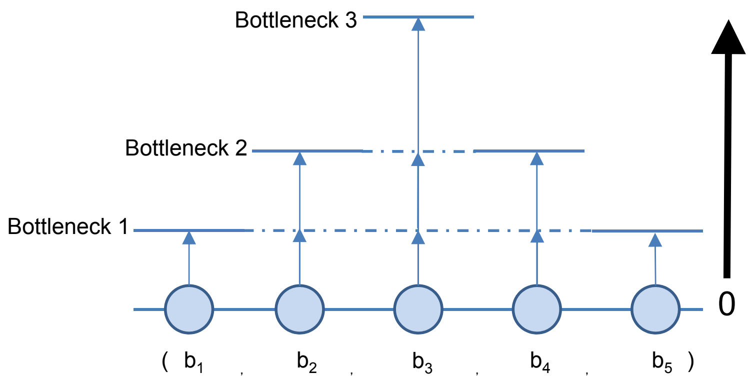

Fig. 1. Scheme of the Bottleneck Flow Control algorithm. Starting from 0, traffic flows are increased until they meet the first bottleneck, as for b_1 and b_5 who are assumed to share a link. The process continues until all traffics are assigned.

To overcome this problem, a different way of assigning traffic by the so-called Bottleneck Flow Control (BFC) has been proposed [3]. This algorithm will be summarized here: the traffic flows (i.e. the state) will be assigned in an iterative manner (see Fig. 1) starting with 0 for all users. Then, the value will steadily increase as long as no sum of traffics at any link exceeds a maximum capacity. But for some traffic flows, this will happen, and we meet a so-called *bottleneck* (as between users 1 and 5 in the figure). So far, we have assigned *the same traffic* to all users, but a further increase of same flows is no longer possible. So, the increase for the users sharing the bottleneck link will be stopped, but it will be continued for all other users (2, 3 and 4 in the example). Then, further increasing the flow will fill another bottleneck (as for users 2 and 4 in the example), and also these users have received the same traffic flows so far. The procedure will continue, until all bottlenecks are filled.

We note that the implementation of the algorithm does not need the "flooding" aspects, as the bottlenecks can be directly computed from the maximum capacities. Nevertheless, the algorithm ensures a non-zero traffic assignment to each user, avoids congestion at any link, and is such that groups of users receive the same traffic flow as long as possible. Thus, the state at the end is considered a *fair* assignment of traffic to users [2]. This will be detailed in the next subsection.

2.2 Fairness relations

We can now consider the maximum fair dominance relation [5] as a rationale for the BFC result: among all states for a given routing, the state x is maxmin fair dominating the state y if and only if for each i with $y_i > x_i$, there is at least one $j \neq i$ such that $x_j \leq x_i$ and $y_j < x_j$. Then, the state assigned by the BFC algorithm is the best element of the maxmin fair dominance relation, and it is also the only element of its maximum set.

Thus, the maxmin fair dominance relation can be used to compare different states for the same routing. However, the same relation can also be used to compare the maxmin fair states for *different routings*, and thus to compare routings. This is the main proposal of this paper. By this rationale, there will also be a maximum set of routings. So, if there is a method for generating different routings, we can consider this maximum set and use it as a means to select a routing in a fair manner.

In this paper, we consider a simple approach to routing variation by employing the Dijkstra algorithm with random weights. Algorithms like the Dijkstra algorithm, or the Floyd algorithm, given a weighted graph, assign a path between each pair of nodes with a minimum sum of weights. If using *random weights*, we will create random paths. Since the paths are related to a minimization criterion, there will be a preference for shorter paths with a small number of hops. For space reason, we cannot provide a detailed analysis of this relation here.

Since maximum sets may have more than one element, the final selection of a single element can be based on a straight optimization point of view. In this paper, we use the selection of the element of the maximum set of all routings with largest sum of components.

3 Fair Routing Path Selection

In the proposed fair routing approach, the final goal is the selection of paths connecting sender-receiver pairs, i.e. the routing. Practically, routing is usually achieved by a shortest path selection procedure. This ensures a minimum number of hops. However, according to a fairness criterion, this might not always be the best choice, since the number of hops is not directly related to link capacity and traffic congestion.

Here, we are focusing on an approach to routing path selection that can take fairness into account, automatically avoids traffic link congestion, and is, as the following experiments show, easy to compute for a localized segment of a network.

We recall the following inferences that can be achieved by the algorithms that were presented in the foregoing section:

1. Given the *routing* (i.e. graph G of the network, sender-receiver pairs P, and their connecting paths), and the maximum capacities c, we can find the maxmin fair state by the BFC algorithm.
2. Given a graph G, sender-receiver pairs P, and weights assigned to each link, we can find a routing by Dijkstra's algorithm by minimizing the sum of weights along the paths connecting senders and receivers.
3. Given a set of states, we can find its maximum set for the maxmin fair dominance relation by direct evaluation.
4. Given a set of states, we can find the element with the largest total sum of throughputs by direct evaluation.

Now we link these inferences together by adding a randomization step, as it is shown in Fig. 2 to achieve the proposed procedure.

Initially, we have a graph G, a set P of sender-receiver pairs, and maximum link capacities for all links in the network.

At first, we generate a set of k random weight assignments to the links of the network. These weights do not have a specific meaning, but they give the base for the Dijkstra algorithm to find (shortest) paths for all sender-receiver pairs, and thus assign a routing, one for each of the k random weights assignments (inference 2).

For each of the k routings, we can find the maxmin fair state by the BFC algorithm, assigning traffic flows to each user, and thus achieving a state for each routing (inference 1). From these states, we can find the maximum set of the maxmin fair dominance relation (inference 3), and from the maximum set, we select the state with the largest sum of components, which is the "throughput maximizing fair state" (TMFS) (inference 4).

After this procedure, given a network, sender-receiver pairs, and maximum capacities, we have generated two things: a routing, i.e. a way of connecting

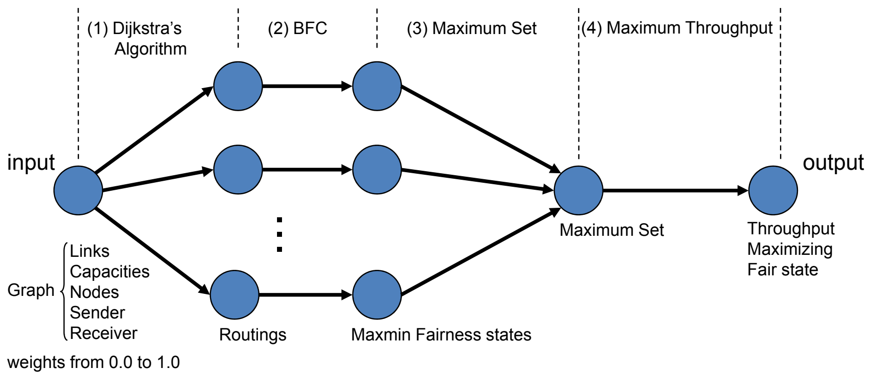

Fig. 2. Inference scheme of the throughput maximizing fair state routing.

senders and receivers in the network (the routing that resulted into the finally selected TMFS) and a state that assigns traffic to each user (the TMFS).

The method would be complete, if we could investigate *all possible routings*. However, the number of possible routings in a network will grow rapidly with network dimensions, and an exhaustive search is impossible with any reasonable computational effort. Therefore, the possible routings are sampled here by assigning random weights. As the used Dijkstra algorithm then is focusing on paths with the smallest sum of weights, the approach nevertheless favors shorter connections, and also avoids any loops in the paths.

4 Experiments

In this section, we will focus on the real-time applicability of the proposed approach. By the rapidly growing number of possible routings in a network, a comparison to the real TMFS is impossible even for low network dimensions. From the specificity of the approach, a direct comparison with other methods is hardly possible as well, as maxmin fairness is not a relation that is maximizing a numerical value.

We might rather ask about a reasonable number of samples (indicated by k in the foregoing section), and how much increasing this number will affect the final outcome of the sampling, i.e. the performance (sum of components) of the TMFS. This was guidance for the design of the following experiments.

4.1 Size of maximum set

In this experiment, the size of the maximum set of the k maxmin fair states was monitored when increasing the number k of states by a computer simulation. Some example results are shown in Fig. 3. The used networks had 10 nodes, were randomly connected by the Barabasi-Albert model [1], and each link had a maximum capacity of 100[1]. At most 30 sender-receiver pairs were randomly selected.

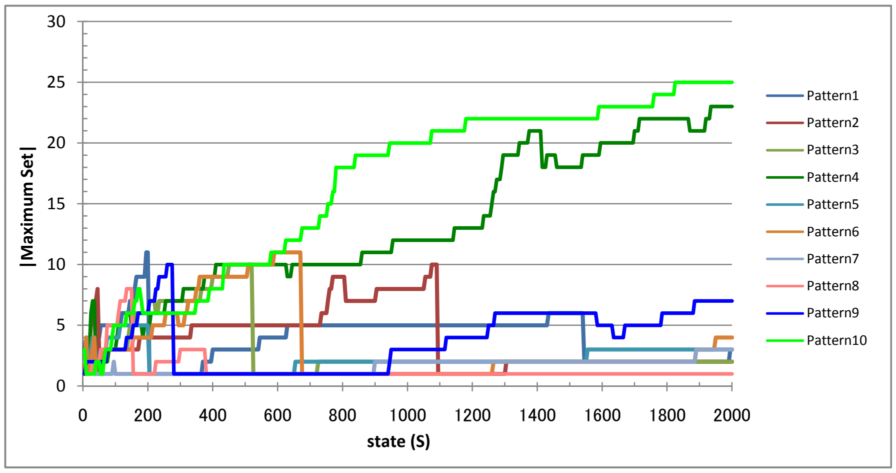

Fig. 3. Size of maximum set vs. number of states for several network configurations.

We can observe a common pattern for the evolution of the size of the maximum set with increasing number of samples: it will nearly linearly increase with the number of samples, but then, often, suddenly drop to a much smaller value. After this, the increase may go on, but with a notably smaller slope. Then, for given network dimension, we often have the strong drop before evaluating 1000 samples. Nevertheless, exceptions like patterns 4 and 10 of the figure are possible.

4.2 Maximum throughput

In addition to the observations from the first experiment, we still cannot know how "far away" from the real maximum set we are, even after evaluating thousands of samples. However, as we are selecting the TMFS at the end, the more relevant question might be how strongly the sum of components of the selected states will vary at all. This was the topic of the second experiment.

In the same configuration as the former experiment, we also computed the maximum throughput (i.e. sum of components) of the maximum states. Figure 4 plots these values for the same cases as before, and it can be clearly seen that

[1] The reason for choosing a constant here is to avoid biases caused by single links with lower capacities that are not related to link sharing. The choice for the value 100 is just for numerical convenience. All used operations here are scale-invariant.

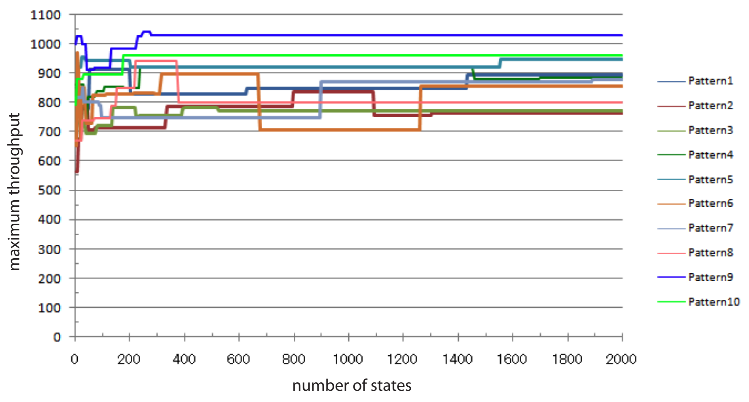

Fig. 4. Maximum throughput vs. number of states for several network configurations.

while the size of the maximum sets can vary greatly, their maximum throughput does not. In each case, we even achieve comparable traffic flows.

This has a strong implication for the practical application of the procedure: first, by using the Dijkstra algorithm, we focus on shorter and cycle-free paths; second, by using the BFC algorithm and the maximum set of its different results, we are focusing on a routing selection with fair traffic assignment; but third, we will achieve a comparable performance even after evaluating a smaller number of samples only.

As a result in the specific case of the experiment (10 nodes, 30 pairs) it can be seen that $k = 500$ samples are sufficient for a fair routing selection with throughput maximization at the end.

4.3 Computational effort

In this section, we want to compare the computational effort needed for the various processing steps of the proposed approach. This is a necessary study in order to evaluate the applicability of the proposed approach under real-time conditions. The following set up was used: number of nodes $|N| = 10$, and the networks were also randomly linked by the Barabasi-Albert model. The maximum capacity of all links were set to 100, and the number of sender-receiver pairs grew from 5 to 50 in steps of 5.

The computation times were measured on a PC with AMD Athlon$^{®}$64 Processor 3200+ (2.00GHz), running Microsoft Windows XP Professional Version 2002 SP3. The PC had 1.00GB of memory, and the programming environment was Java 1.6.0_23. The results can be seen in Fig. 5. There, the time in μs needed to find p_{max} versus number of pairs used in the sampling is shown.

It can be seen that there is a linear increase in total processing time, as well as for the main components "BFC" (largest share of processing time) and "Dijkstra's Algorithm" (the second largest share). Note that the processing time

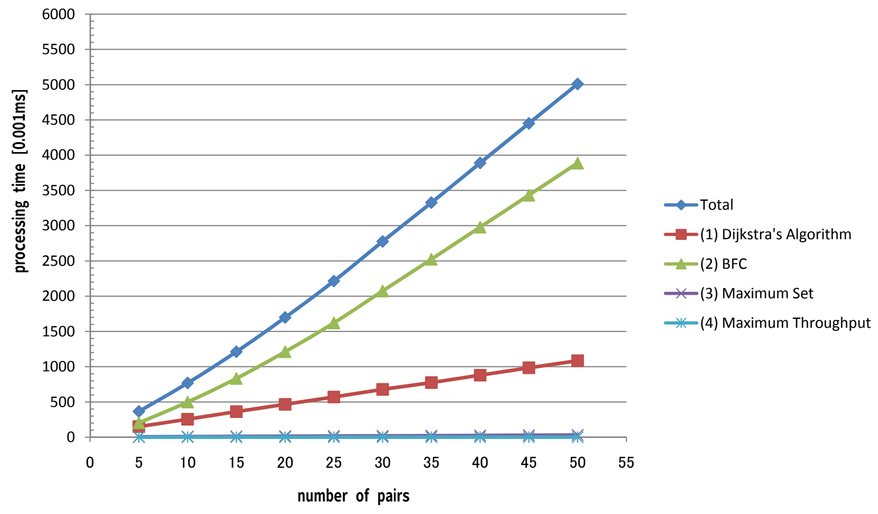

Fig. 5. Shares of processing time vs. number of pairs.

for Dijkstra's Algorithm increases differently for the number of nodes, but here, we are focusing on the number of pairs that send traffic through a network for a fixed architecture.

The other shares are barely visible in the graph, in fact, the share for finding the maximum set, and for selecting the largest throughput from the elements in the maximum set can be neglected in comparison to the processing time needed for the other steps.

5 Summary

The uncontrolled sharing of network resources can easily lead to unwanted situations for collaborating users, especially traffic congestion and response delays. Often, the reasons for this are not transparent to the user (for example, the user has no direct sensation of a "bottleneck" in a network, or the way her or his data packets are routed). In fact, there are no objective bottlenecks in a network, as present analysis has shown that a bottleneck is a virtual concept of congestion models. The approach presented in this paper is suitable to report direct information to the user that can be used to improve the efficiency of shared network employment for all users together. The approach is based on fairness, more specific on the selection of the element with the largest total throughput from the maximum set of the maxmin fair dominance relation for a probing sample of random routings.

Once we have obtained this state, the further processing is open to some choices. These days, routers (i.e. the nodes in the network graph) are not "intelligent" and do only what their name indicates: route the traffic, and drop overexcess traffic packages. The fair traffic rates can be implemented by router communication, but they can be negotiated between users as well. The main desiderata here is that the user will realize that any attempt to increase traffic in current network situation will cause other users (of her or his group) to experience worse performance - this is ensured by the maxmin fairness definition.

The reported experiments demonstrate that, for example, for a group of about 10 users, 500 samples are sufficient to report a suitable fair traffic allocation to users. Future work will focus on a more efficient sampling method, based on meta-heuristics, and the extension of the concept by including the trade-off with delay time in the relational approach. Also, we will have to provide more efficient methods to generate representative routing samples with regard to a following fairness evaluation.

Acknowledgment

This work was partly supported by the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (S) (No. 18100001).

References

1. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74, 47–97 (Jan 2002)
2. Bertsekas, D., Gallager, R.: Data Networks. Englewood Cliffs, N: Prentice Hall (1992)
3. Jaffe, J.: Bottleneck flow control. IEEE Trans. Commun. COM-29 (July 1981)
4. Kelly, F.: Charging and rate control for elastic traffic. Eur. Trans. Telecomm. 8, 33–37 (Jan/Feb 1997)
5. Köppen, M., Tsuru, M., Oie, Y.: Evolutionary approach to maxmin-fair network-resource allocation. In: Applications and the Internet, IEEE/IPSJ International Symposium on. pp. 253–256. IEEE Computer Society, Turku, Finland (2008)
6. Mo, J., Walr, J.: Fair end-to-end window-based congestion control. IEEE/ACM Trans. on Networking pp. 556–567 (2000)
7. Nakamura, K., Tsuru, M., Oie, Y.: On the framework for network measurement as a service – the perfsonar-based integrated network management. In: Intelligent Networking and Collaborative Systems, International Conference on. pp. 325–326. IEEE Computer Society (2010)
8. Suzumura, K.: Rational Choice, Collective Decisions, and Social Welfare. Cambridge University Press (2009)
9. Verschae, R., Köppen, M., Yoshida, K.: Partial user-supplied information and user modeling for improving QoS. Simulation Modelling Practice and Theory 19(1), 47–55 (January 2011)