

12. IP Routing Algorithms

Mirosław Dynia
mdynia@upb.de
Immatriculation Nr.: 6240730
and
Mirosław Korzeniowski
rudu@upb.de
Immatriculation Nr.: 6202823

University of Paderborn

Organizer: Christian Schindelhauer

August, 4th, 2004

1 Introduction

In this survey we describe and examine routing problem in the internet from both practical and theoretical points of view. As for practice, we describe the existing algorithms that take care for building routing tables. On the theoretical side we describe a few areas of research. Those are oblivious routing, packet switching and nash equilibria.

2 IP Routing in Real Life

What really happens in the real world? We are going to describe a few algorithms which maintain routing in the Internet.

Routing involves two activities: determining routes and transporting packets. The first activity is the crucial one and rely on maintaining routing tables placed in routers. Transporting packets is simple algorithm using information from routing table. It is just switching packet from one router to another, closer to packets destination. This switch operation is repeated until packet reach destination or meet its delivery deadline (so called Time To Life).

Now we will present model of Internet and algorithms which maintain routing tables.

2.1 Model of the Internet

The Internet architecture distinguishes *hosts* and *gateways*. Hosts are computers that execute application programs on behalf of users. Gateway connects two or more network into internets. Gateways (routers) own a routing table maintained by certain routing algorithms. An **autonomous system** (AS) is a network or group of networks under a common routing policies.

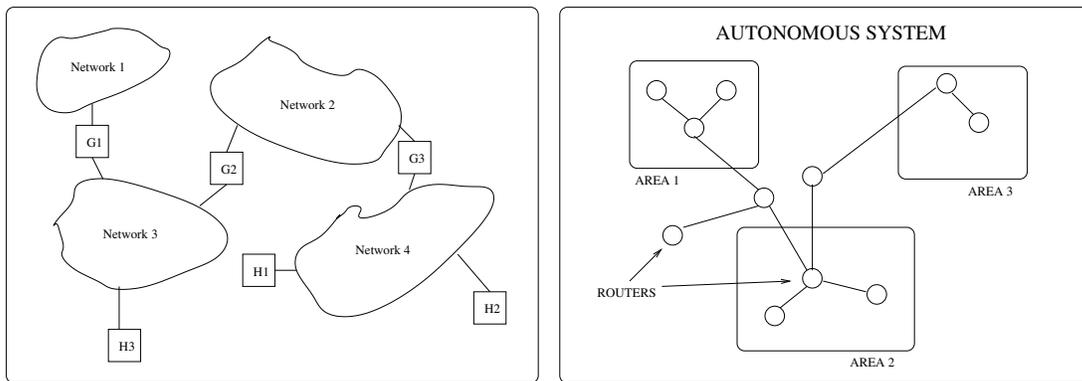


Figure 1: Model of the Internet.

2.2 Switching

Switching process is relatively simple, and is the same for most routing protocols. When new packet (let say with destination A) appears in the network then it is being sent to the nearest router. Router checks routing table and forward packet to the next hop, Otherwise, if there is no information in routing table about next hop, router drops the packet. For one packet the route appears to be static, fully described by routing tables. From the other point of view there are many changes in the network, and these should be reflected in the content of the routing table. There are many algorithms which updates routing tables, so that paths in switching process are chosen with respect to the situation in the network.

2.3 Algorithm's overview

There are many algorithms which handle routing in networks. Nice overview and RFC links can be found in [oC01].

Routing tables are meant to reflect network topology in any point of time. Many algorithms are implemented in order to achieve this goal. They can be split into two groups. "Link-status" protocols propagate the status of the actual topology (connections, congestion). Every router builds in routing tables picture of whole network. "Vector distance" protocols exchange simply parts of the routing tables of the neighboring nodes. In essence link-state algorithms send small packets everywhere, while distance-vector algorithms exchange large parts and only with close neighborhood.

Border Gateway Protocol BGP is a "vector-distance" protocol used to exchange routing information and is the protocol used between autonomous systems (e.g. Internet Service Providers).

Interior Gateway Protocol IGP is a **class** of protocols (including RIP, OSPF, IGRP, EIGRP, IS-IS) for exchanging routing information within AS (e.g. customer networks, universities and corporations).

2.4 Routing Information Protocol

There had been many versions of this protocol developed to cover changing environment of the Internet. The RIP was developed as a part of BSD UNIX at the University of California mid-'80. It is also confusing, that many similar protocols has been called RIP protocols. We describe extended version of RIP (it is RIP 2) algorithm described in November 1994 by [Mal94].

This is a vector distance protocol which uses only one simple "number of hops" metric to measure distances. Router *A* sends to its neighbors approximately every 30 sec. part of the routing table. Lets say any other router *B* gets information (*C*, *h*). Router *B* takes route to *C* through router *A* only if it is better one (number of hops *h* is smaller than for route that it learned before). Moreover *B* refuse information with $h \geq 16$ which prevents loop situation. This technique has very good advantage - simplicity, but unfortunately suffers from a quite poor convergence.

2.5 Border Gateway Protocol

Specification of the BGP is contained in [RL95]. There is a bit of confusion concerning name of this protocol. If BGP is to exchange routing information between ASs then it is called **External BGP** and if it is used for routing inside ASs then its called **Interior BGP**. When BGP environment detect a new neighbor then they exchange full routing information. After that changes in routing tables causes exchange only of parts of routing tables between neighbors. Routers do not send periodic routing updates.

There are several attributes assigned to each BGP router. One of these is *Weight*, and is defined by Cisco and reflect importance of router. When two routers *A* and *B* advertise for network *X* to router *C*, then *C* save both bit of information in BGP routing tables, but only one router with bigger *Weight* will be set as next hop in IP routing table.

In order to chose the best one from advertised routes many cases are being checked. First 3 of them are:

- If the path specifies a next hop that is inaccessible, drop the update.
- Prefer the path with the largest weight.
- If the weights are the same, prefer the path with the largest local preference.

An order order of this list is important for the protocol performance.

2.6 Open Shortest Path First

The OSPF protocol [Moy91] was developed due to growing incapability of the RIP protocol of serving large, heterogeneous networks. It is based on Shortest Path First algorithm. The link-state messages are sent along the network. Each router collects this link-state advertisement (LSA) and uses as input to our algorithm. Two routers are said to be adjacent if their link-state databases are synchronized. Routing proceeds only in the set of adjacent routers.

The OSPF works within a hierarchy. The largest entity is an autonomous system which is divided into several areas. Routers within the same area share the same topological database. If there are some routers at the border of area, they are called Area Border Routers. These maintain topological databases of other bordering areas.

Given information gathered by routers from LSAs, a building of shortest-path tree is possible. Each router is in the root of his own tree, which can be updated very quickly in the face of changes in the network. The Shortest-path tree serves as set of routing paths. If destination host is within the same area, then finding path is easy, otherwise packet is being send to ABR which knows route to destination area.

2.7 Intermediate System to Intermediate System (IS-IS)

Integrated IS-IS [Cal90] is a link-state routing protocol that floods the network with link-state information, to build complete, consistent picture of network topology. There are two terms concerning this protocols: Intermediate System (gateway) and End System (host). Moreover this entities are merged by system administrator into areas. ISs inside particular area are called *Level 1 ISs* and learn only topology inside this area. *Level 2 ISs* learn positions of other *Level 2 ISs* and make possible routing between areas. Basically all routes are learned by sending HELLO messages.

Sending packet between ESs is easy and run in the following way. The packet from ES goes to the *Level 1 IS* which knows position of *Level 2 IS*. Since *Level 2 IS* is in the backbone, it knows route to ES of destination.

This protocol form a similar hierarchy to the OSPF protocol, but works in much simpler way. Since it does not build tree for each router it is

2.8 Interior Gateway Routing Protocol

This distance vector algorithm was developed in the mid-1980s by Cisco Systems. Since this protocol is more robust and capable for large networks it replaced RIP protocol which serves routing only for small networks. The IGRP provides new stability features as hold-downs, split horizons and poison-reverse updates. There are update messages send every 90 seconds. When one router breaks down, then other router tries to advertise this in the network. The hold-downs prevent the network overflow. Split horizon and poison-reverse are protections against loops in routing tables.

Nice feature of IGRP is allowing multi-path routing. Protocol handle more than one route to one destination and in a round-robin fashion replace entries in routing tables. The network administrator can influence this process and set variance which describe which routes (in which cost interval) are candidates for multi-path routing. Through evolution a new protocol was developed. The Enhanced IGRP includes some additional features that through some additional structures improves performance of IGRP.

3 Theory on Routing

In this section we deal with two theoretical aspects of routing. First we show some results on so called oblivious routing which is an area that had real breakthrough in the recent years. The main issue in this manner is to choose the paths along which packets will travel. In oblivious routing

those paths are chosen independently from one another and there is no need for synchronization or central management of the network except for some precomputation.

Further on we concern ourselves with what happens with the packets that are injected into the network. We present the area of reaserch that analyzes the routines which decide which packets should go first when many packets compete for the same edge.

In both of these areas we will be interested in those results that are as general and distributed as possible. Therefore we are not describing the offline centralized algorithms or results for specific topologies too deeply.

3.1 Oblivious Routing

In this subsection we describe results on oblivious routing. First we define what a routing problem is and outline the difference in the notions of offline centralized routing and oblivious routing schemes.

Assume we are given a graph $G = (V, E)$ with a function $c : E \rightarrow \mathbb{N}$ describing capacities of the edges. A routing problem on G is a set \mathcal{D} of demands (s_i, t_i, d_i) , meaning "send d_i units of data from s_i to t_i ".

The solution can be fractional, that is it can arbitrarily divide any commodity that has to be sent. This means that we do not have to choose one path for each demand but can rather choose a flow from s_i to t_i . In the real network it can then be interpreted as choosing one of those paths with probability proportional to the fraction of the demand that should be sent along it. In the solution for such a routing problem we need to optimize congestion \mathcal{C} . The latter is defined as follows.

For any edge $e \in E$, congestion on e is defined as the sum of the demands that use this edge. The congestion \mathcal{C} of the whole network is the maximum of the congestion of all the edges.

Now we come to the difference between the formulation of the offline and oblivious program. In both formulations the graph is arbitrary and is not a part of the input. The offline algorithm can first input the routing problem and then output the solution, whereas the oblivious algorithm has to output the solution without reading the problem. This means that the oblivious algorithm has to choose systems of paths for all possible pairs $(s, t) \in V \times V$ so that no matter what problem is given as an input it can be routed using the given systems of paths and the produced congestion should still be low.

The latter can seem unfair for the oblivious program but we will see that it is possible to design oblivious schemes that will behave not much worse than the optimal offline solution. We say that an oblivious scheme for graph G is q -competitive if for every possible routing problem the congestion produced by this oblivious scheme is at most q times larger than the congestion of the optimal offline solution.

Further on we show that there are oblivious routing schemes that are $\text{polylog}(n)$ -competitive (where n is the number of nodes in G) and that they can be computed in time polynomial in n .

One of the first important results on oblivious routing is [MMVW97]. One of the results in this paper is that for a 2-dmensional mesh optimal oblivious routing has competitive ratio $\Theta(\log n)$. The authors showed an algorithm which achieves such competitive ratio and proved lower bound for this problem by showing an adversary which forces any oblivious scheme to produce congestion which is by a logarithmic factor higher than optimum.

Theorem 1 *For a 2-dimensional $m \times m$ mesh and any oblivious routing scheme on the mesh there exists a routing problem f such that the optimal solution for f achieves congestion by a factor of $\Omega(\log m)$ smaller than the oblivious routine.*

In a seminal paper from 2002 [Rae02], Harald Raecke proved that it is possible to design an oblivious routing scheme that is $\mathcal{O}(\log^3 n)$ competitive, provided that much time and work can be spent on precomputation. The latter limitation was coped with in [BKR03, HHR03] where it was proved that it is possible to pseudopolynomially bound the time spent on precomputation and still achieve polylogarithmic competitive ratio. The competitive ratios in those papers

were $\mathcal{O}(\log^4 n)$ and $\mathcal{O}(\log^2 n \log \log n)$, respectively. The time for precomputation is described as pseudopolynomial due to the fact that it is polynomial in n and $\max_e \{c(e)\}$.

In [ACF⁺03] it was shown how to precompute an oblivious scheme with the best possible competitive ratio in polynomial time. The drawback of this paper is that it does not state the result explicitly. One has to use the result from [HHR03] to say that the competitive ration achieved is $\mathcal{O}(\log^2 n \log \log n)$. Besides, this solutin uses the ellipsoid algorithm with separation oracle wich makes things complicated for implementation and the running time is very high.

Tree decomposition

Here, we deal with the idea behind [Rae02, BKR03, HHR03]. All three papers work on laminar tree decomposition of the graph G . The root of the tree represents the set of all nodes of G . The children of each node represent sets which form partition of the set represented by their father and leaves of the tree represent sets consisting of single nodes. Additionally it is assured that the size of every child's set is at most a constant fraction of the size of father's set. This guarantees that the depth of the tree is logarithmic in n .

The process of dividing a single set into subsets approximates the optimal solution to a Concurrent Multicommodity Flow problem (which is essentially a routing problem) defined according to the capacities of edges leaving the main cluster and to the capacities crossing the already chosen subdivision. If the congestion of the solution is low, the current iteration is done and the computation for newly computed children can begin. In other case the algorithm finds a conuterexample proving that no sufficiently good solution can be found for so defined flow problem. This counterexample is a sparsest cut which can be then used to further divide the current clustering.

The actual routing scheme uses the tree in the following manner. A path from node u to node v will use all the sets on the unique path from the leaf representing $\{u\}$ to the leaf representing $\{v\}$ in the tree. In each of the sets on this path two intermediate nodes are chosen randomly and the paths between them are chosen according to the solution of the flow defined at the stage of building the tree. One of those nodes is chosen with probability distribution depending on the division of the set into smaller sets and the other with probability distribution depending on the connections of nodes to the father in the tree. For lower bound on the congestion of the optimal algorithm it is shown that the oblivious algorithm sends something outside a set in the tree if and only if the optimal does. This together with logarithmic height of the tree and polylogarithmic factors for approximation of concurrent multicommodity flows and sparsest cuts yields polylogarithmic competitive ration for the oblivious scheme.

The Best Oblivious Algorithm

Below, we shortly sketch the solution from [ACF⁺03]. The authors state the routing problem as a Linear Programming problem. It expresses the condition that the oblivious solution has to be q competitive for any demand-matrix, where q is the variable to be minimized. The part "any matrix" implies that the set of constraints is infinite but this is not a problem when using ellipsoid algorithm. The latter works in the following fashion: it produces some solution which has to be evaluated. After the evaluation process we know that the produced solution satisfies the constraint that the competitive ratio is q for any given demand-matrix or we can find a demand-matrix which contradicts it. This matrix is then used by the ellipsoid algorithm to improve the solution.

The theory of LP-solving and the ellipsoid algorithm with separation oracle in particular, guarantee that the running time and the number of iterations will be polynomial in the description size of the graph G , i.e. the number of nodes n and the size of the bit representation of edge capacities.

3.2 Packet Switching

In this part we concentrate on a different aspect of routing, namely queuing theory. When all the routes for all the request have been chosen there still remains one problem. Even if the paths have

been chosen optimally we still have to decide what exactly should be done when at some point in time two packets want to use the same edge. One of them can proceed, the other has to be delayed.

Our model for the network will be as follows. The network itself is a directed graph (any undirected graph can be depicted as a directed graph with double links). Time is divided into discrete steps and in each time-step any edge can be crossed by at most one packet. The problem is to choose such policy for the packet switching that minimizes the time when the last packet reaches its destination. We assume that packets are routed along simple paths, i.e. no edge is used twice by the same packet.

Each (directed) edge has a buffer at its beginning - this buffer stores all the packets that have reached this edge and want to cross it. Besides optimizing the time to finish the routing we also want to keep the size of any buffer small.

We say that a policy is greedy if it always forwards some packet along each edge that has non-empty buffer. It is trivial that each non-greedy policy can be transformed into a greedy one such that, for any problem, the non-greedy strategy finishes not later than the greedy one.

It is easy to see that the routing time is always $\Omega(\mathcal{C} + \mathcal{D})$ where \mathcal{C} denotes the congestion and \mathcal{D} denotes dilation produced by the solution. On the other hand any greedy policy achieves time $\mathcal{O}(\mathcal{C} \cdot \mathcal{D})$. We state after [LMR88], [LMR94] and [LMR96] that it is possible to compute offline a schedule that is optimal up to constant factor. A good survey of this area of research can be found in [Sch98]

Theorem 2 *For any set of packets with simple paths having congestion \mathcal{C} and dilation \mathcal{D} , there is an offline schedule that needs $\mathcal{O}(\mathcal{C} + \mathcal{D})$ steps to route all packets, using buffers of constant size. Moreover, such a schedule can be found in time $\mathcal{O}(P \cdot \log \log P \cdot \log P)$, where P is the sum of the lengths of all paths.*

However, a centralized offline algorithm is only a beginning. Similarly as in the previous section we would like to concentrate on protocols that would fit reasonably to internet, that is protocols that are on-line and local. Much work has been done in this area. We will shortly state most of those result and present in details only one proof as an example.

We consider the following adversarial model introduced by [BKR⁺01].

At each step the adversary generates a set of new packets and paths that they have to travel. We concentrate on the case when the paths cannot be changed later, i.e. nonadaptive routing. We introduce the following restriction on the adversary. Let w be an arbitrary positive integer, e any edge in the network and τ any sequence of w consecutive time steps. We define $N(\tau, e)$ to be the number of paths injected by the adversary during time interval τ that traverse edge e . For any $\rho > 0$, we define a (w, ρ) adversary that injects paths subject to the following load condition: for every sequence τ of w consecutive time steps and every edge e , $N(\tau, e)/w \leq \rho$. We say that such a (w, ρ) adversary injects packets at rate ρ with window size e . A rate ρ adversary is a (w, ρ) adversary for some w .

Such a restriction of the adversary, essentially means that in time interval of length t , the adversary can generate at most $\lceil rt \rceil$ requests that use the same edge. We use a little broader class of adversaries introduced in [AAF⁺01]. The rate of the adversary will be specified by a pair (b, r) where $b \geq 1$ is a natural number and $0 < r < 1$. In any interval I , the adversary can inject at most $r \cdot |I| + b$ packets that use the same edge. The parameter b allows the adversary to use bursty patterns of injection.

Our main goal is to prove stability (or instability) of different scheduling policies in different networks. We say that a given scheduling policy \mathcal{P} is stable against a given adversary \mathcal{A} for a given network G if for any initial configuration of packets in the network $C_0(G)$ there is a constant M (depending on the size of the network G , the initial configuration, the rate (b, r) of the adversary \mathcal{A}), such that when the system is executed with the initial configuration $C_0(G)$ against adversary \mathcal{A} , the maximum number of packets at any time step in any queue is bounded by M .

This definition of stability is equivalent to a definition in which we bound the total number of packets in system rather than the number of packets in a single buffer. One could also define the

stability in terms of packet delay (which is the number of time steps that a packet has to spend in queues) rather than the queue size but for $r < 1$ those definitions are equivalent.

We say that a network G is universally stable with respect to a class of adversaries Y if every greedy scheduling policy is stable against every adversary \mathcal{A} from Y for the given network. Similarly, we say that a given policy is universally stable with respect to a class of adversaries Y , if it is stable against this class for any network.

Further on we concentrate on stability results for the following scheduling policies.

- FIFO (First-In-First-Out) gives priority to the packet that has been in this queue for the longest period of time. FIFO is sometimes called FCFS (First-Come-First-Served).
- LIFO (Last-In-First-Out) gives priority to the packet that has been in this queue for the shortest period of time.
- NTG (Nearest-To-Go) gives priority to the packet which has the smallest number of edges still to be traversed.
- FFS (Farthest-From-Source) gives priority to the packet which has traversed the largest number of edges.
- FTG (Farthest-To-Go) gives priority to the packet which has the largest number of edges still to be traversed.
- NTS (Nearest-To-Source) gives priority to the packet which has traversed the smallest number of edges.
- SIS (Shortest-In-System) gives priority to the youngest packet.
- LIS (Longest-In-System) gives priority to the oldest packet.

Below we state a few results concerning some specific networks. The first three are taken from [BKR⁺01]. First result deals with any directed acyclic network.

Theorem 3 *Every DAG (directed acyclic graph) is universally stable against the class of rate 1 deterministic adversaries.*

The next two results show that ring topology is unstable for some schedules and stable for others.

Theorem 4 *LIS and FIFO are not stable against the class of rate 1 deterministic adversaries for the ring network.*

Theorem 5 *FTG is stable against the class of rate 1 deterministic adversaries for the ring network.*

The last result comes from [AAF⁺01]

Theorem 6 *The ring network is universally stable against the class of rate $r < 1$ deterministic adversaries.*

In the table below, we show which of the protocols mentioned above are stable. For those protocols we also state the bounds on the possible buffer sizes and delays. All the results come from [AAF⁺01]. The notation in the table is as follows: m is the number of edges in the graph, d is the longest path a packet has to traverse and $\epsilon = 1 - r$ is the "clearance" that the algorithm has against the adversary.

<i>Protocol</i>	<i>Queuesize</i>	<i>Delay</i>
<i>FTG</i>	$\Theta(bm^{d-1}/\epsilon)$	$\Theta(bm^{d-1}/\epsilon)$
<i>NTS</i>	$\Theta(bm^{d-1}/\epsilon)$	$\Theta(bm^{d-1}/\epsilon)$
<i>SIS</i>	$\Theta(b/\epsilon^d)$	$\Theta(db/\epsilon^d)$
<i>LIS</i>	$\mathcal{O}(b/\epsilon^d)$	$\mathcal{O}(b/\epsilon^d)$

In the next table we summarize the protocols that are not universally stable. Here, some explanation is needed. The protocols LIFO, NTG and FFS are not stable for adversaries of rate $r \geq 1/\sqrt{2}$. FIFO was originally [AAF⁺01] proved to be unstable for $r \geq 0.85$. This result was later improved to $r \geq 0.8357$ in [DKN⁺01]. This result was then improved to arbitrarily low r in [BG03]. The latter result produced graphs with number of nodes polynomial in $1/r$. On the other hand in [LPSR04] it was proven that FIFO is always stable if $r < d$ where d is the length of the longest path in the system. To the best of our knowledge, the results for LIFO, NTG and FFS have not been improved.

<i>Protocol</i>	<i>Rate</i>
<i>FIFO</i>	0
<i>LIFO</i>	$1/\sqrt{2}$
<i>NTG</i>	$1/\sqrt{2}$
<i>FFS</i>	$1/\sqrt{2}$

Below we give the proof of the classical result (even though it was improved a few times) from [AAF⁺01].

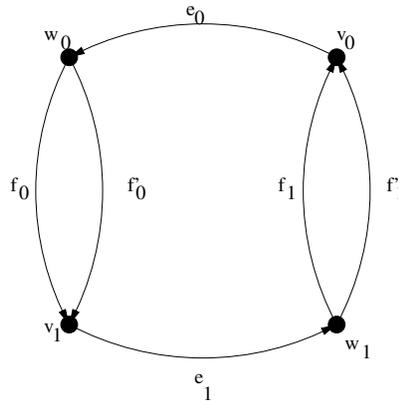


Figure 2: Graph G used to show instability for FIFO

Theorem 7 *Let $r \geq 0.85$. For the graph G depicted in figure 3.2 there is a nonempty initial configuration and an adversary \mathcal{A} of rate r , such that FIFO is unstable against \mathcal{A} in G .*

Proof. The adversary works in phases. The induction hypothesis is as follows: at the beginning of phase j there are at least $s_0 + j$ packets in the queue of e_i , for large enough constant s_0 and $i = 0$ or 1 , depending on whether j is even or odd.

In the beginning there are s_0 packets queued at v_0 and so the induction hypothesis for phase 0 is trivially met.

Assume that j is even and that there is a set S of s packets waiting in the queue of e_0 . The sequence of injections in phase j is as follows.

1. For the first s steps, we inject a set X of rs packets with route $e_0 f'_0 e_1$. These are blocked by the packets in S .
2. For the next rs steps we inject a set Y of $r^2 s$ packets with route $e_0 f_0 e_1$. These are blocked by the packets in X .

We also delay the flow of packets in X through f'_0 using single-edge injections. The new packets get mixed with the packets in X . In the process $rs/(r+1)$ packets of X cross f'_0 and the size of X shrinks to $r^2 s/(r+1)$.

3. For the next $r^2 s$ steps the packets in X and Y move forward, and merge at v_1 . At the same time, $r^3 s$ new packets are injected to edge e_1 . Since $r^2 s$ packets cross e_1 , after $r^2 s$ steps the queue of e_1 contains $r^3 s + r^2 s/(r+1)$ packets.

This ends phase j . Since $r^3s + r^2s/(r + 1) > s$, we have at least $s + 1$ packets for phase $j + 1$ waiting in e_1 . ■

In [AAF⁺01] a randomized algorithm was proposed which is stable and its queue sizes and delays are bounded polynomially. The idea behind this algorithm is that each packet is assigned a label at the moment at injection t . This label depends on t and some random λ . Then, a packet with the smallest label is given priority and after forwarding a packet its label is increased by 1.

3.3 Selfish Routing

In this scenario we assume that there is no general regulations concerning routing in network. Every packet tries to chose best route for himself, without worrying about routing time of other packets. Such a scenario is called *selfish routing*. Lets assume that all packets chosen its routes in some (selfish) way. The state where no selfish packet *will* change its route (and make its routing time shorter) is called *Nash equilibrium*. It is well known that Nash state is not always optimal (subject to Total Latency - sum of all travel times). Moreover there could many Nash Equilibrium exist. The ratio between Worst Case Nash Equilibrium and the Social Optimum is called *Price of anarchy*.

There are a lot of works on bounding price of anarchy. Model of unregulated traffic requires a good mathematical description, since int this case intuitions fail very often. The Braess's Paradox describe situation where putting an additional edge to the graph increases price of anarchy. This means that building new 'road' is not always followed by improving overall traffic situation.

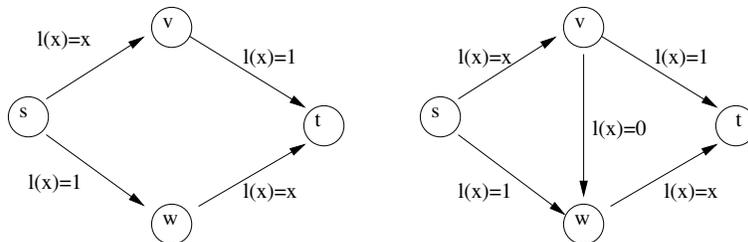


Figure 3: Braess's Paradox.

Lets assume that our network is a graph for which each edge has some latency which depends on congestion of this edge, and packets travel only from s to t . For the left graph the selfish behavior leads to Nash equilibrium and causes latency of $\frac{3}{2}$. On the right picture there is one additional edge with 0 latency. Surprisingly this rise value of latency up to 2.

In [RT00] showed that if the latency of each edge is a linear function of its congestion, then the Total Latency of the routes chosen by selfish network users is at most $\frac{4}{3}$ times the minimum possible Total Latency.

Moreover they show that if we consider smaller restrictions on latency function and assume it to be continuous and nondecreasing in the edge congestion, then price of anarchy can be arbitrary large.

Artur Czumaj and Berthold Voking studied KP-model of network. This is simple graph of 2 nodes with m links between them and different speeds for every link. They showed that for such scenario Price of Anarchy is

$$\Theta\left(\frac{\log m}{\log \log \log m}\right)$$

This tight bound resolves an open question posted by authors of this model Koutsoupias and Papadimitriou.

References

- [AAF⁺01] Matthew Andrews, Baruch Awerbuch, Antonio Fernandez, Tom Leighton, Zhiyong Liu, and Jon Kleinberg. Universal-stability results and performance bounds for greedy contention-resolution protocols. *J. ACM*, 48(1):39–69, 2001.
- [ACF⁺03] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Raecke. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 383–388. ACM Press, 2003.
- [BG03] Rajat Bhattacharjee and Ashish Goel. Instability of fifo at arbitrarily low rates in the adversarial queueing model. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 160. IEEE Computer Society, 2003.
- [BKR⁺01] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queuing theory. *J. ACM*, 48(1):13–38, 2001.
- [BKR03] Marcin Bienkowski, Mirosław Korzeniowski, and Harald Raecke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 24–33. ACM Press, 2003.
- [Ca90] R. W. Callon. RFC 1195: Use of OSI IS-IS for routing in TCP/IP and dual environments, December 1990. Status: PROPOSED STANDARD.
- [DKN⁺01] Josep Diaz, Dimitrios Koukopoulos, Sotiris Nikolettseas, Maria Serna, Paul Spirakis, and Dimitrios M. Thilikos. Stability and non-stability of the fifo protocol. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 48–52. ACM Press, 2001.
- [HHR03] Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 34–43. ACM Press, 2003.
- [LMR88] Frank Thomson Leighton, Bruce M. Maggs, and Satish Rao. Universal packet routing algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 256–271, October 1988.
- [LMR94] Frank Thomson Leighton, Bruce M. Maggs, and Satish Rao. Packet routing and job-shop scheduling in $O(\text{congestion} + \text{dilation})$ steps. *Combinatorica*, 14(2):167–180, 1994.
- [LMR96] Frank Thomson Leighton, Bruce M. Maggs, and Andrea W. Richa. Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules. Technical Report CMU-CS-96-152, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, July 1996.
- [LPSR04] Zvi Lotker, Boaz Patt-Shamir, and Adi Rosen. New stability results for adversarial queuing. *SIAM J. Comput.*, 33(2):286–303, 2004.
- [Mal94] G. Malkin. RFC 1723: RIP version 2 — carrying additional information, November 1994. Obsoletes RFC1388. Obsoleted by RFC2453 . Updates RFC1058. Updated by RFC2453, STD0056. Status: DRAFT STANDARD.
- [MMVW97] Bruce M. Maggs, Friedhelm Meyer auf der Heide, Berthold Voecking, and Matthias Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, page 284. IEEE Computer Society, 1997.

- [Moy91] J. Moy. RFC 1247: OSPF version 2, July 1991. See also RFC1246, RFC1245 . Obsoleted by RFC1583. Obsoletes RFC1131 . Status: DRAFT STANDARD.
- [oC01] Web Pages of CISCO. http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/. 2001.
- [Rae02] Harald Raecke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 43–52. IEEE Computer Society, 2002.
- [RL95] Y. Rekhter and T. Li. RFC 1771: A Border Gateway Protocol 4 (BGP-4), March 1995. Obsoletes RFC1654. Status: DRAFT STANDARD.
- [RT00] Tim Roughgarden and Eva Tardos. How bad is selfish routing? In *IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2000.
- [Sch98] Christian Scheideler. *Universal Routing Strategies for Interconnection Networks*. Springer-Verlag New York, Inc., 1998.