6 Adversarial Switching

So far, we only looked at *static* routing problems, i.e. all requests are given from the beginning and no new request arrives during the routing. In the real world, however, requests usually arrive at the nodes in a continuous fashion. Models that take this into account are called *dynamic routing models*, as opposed to the *static routing models* we considered before. These models can be usually characterized as either *stochastic* or *adversarial*. In stochastic models, the injection of requests is modelled with the help of stochastic processes, whereas in the adversarial model it is assumed that an adversary controls the injection of new requests. In this section, we restrict ourselves to considering the following adversarial model.

6.1 The adversarial injection model

In this model we have an adversary that is allowed to demand network bandwidth up to some specified limit. That is, it is allowed to choose any node at any time step to inject a new packet and it is allowed to select any path for the injected packet as long as the load of the edges does not exceed a certain limit. More formally, for any $w, \lambda > 0$, an adversary is called a *bounded adversary of rate* (w, λ) if for all edges e and all time intervals I of length w, it injects no more than $\lambda \cdot w$ packets during I that contain edge e in their routing paths. λ is called the *injection rate* and w is called the *burstiness* of the adversary. We demand that the adversary has to tell the system the paths it selects. Thus, we are left with solving a *dynamic switching problem*. An algorithm for this problem is called a *dynamic switching protocol*. Such a protocol is called *stable* for some λ if for any injection rate of at most λ and any network the (expected) number of packets in the system does not grow unboundedly with time. Since an edge can transport at most one packet per step, λ can be at most 1. A protocol that is stable for any $\lambda < 1$ is called *universally stable*. We will show that there are both universally stable and not univerally stable protocols.

6.2 Universal stability of SIS

The SIS protocol (which is a short form of "shortest-in-system") always gives preference to the youngest packet. Ties may be broken arbitrarily. For this protocol we can prove the following amazing result.

Theorem 6.1 *SIS is universally stable.*

Proof. Let $0 < \epsilon < 1$ be chosen such that $\lambda = 1 - \epsilon$. Suppose that there is a packet P that requires more than $\sum_{i=1}^{d}(w+1)/\epsilon^i$ steps to traverse a path of length d. In this case there must be an $i \in \{1, \ldots, d\}$ for which P was delayed for at least $(w+1)/\epsilon^i$ steps at the ith edge of its path. Consider the minimal i for which this holds. Let e be the corresponding edge on P's path. Then the time difference T between the injection of P and the time at which P had to wait for the $(w+1)/\epsilon^i$th time at e is at most

$$\sum_{j=1}^{i}(w+1)/\epsilon^j = \frac{w+1}{\epsilon} \cdot \frac{(1/\epsilon)^i - 1}{(1/\epsilon) - 1} = (w+1) \cdot \frac{(1/\epsilon)^i - 1}{1 - \epsilon} .$$

Since $\lambda = 1 - \epsilon$, at most

$$(1-\epsilon)T + w = (w+1) \cdot \Big((1/\epsilon)^i - 1\Big) + w < (w+1)/\epsilon^i$$

packets can be injected during T time steps that intend to cross e. Only these packets can be preferred against P at e. However, since our assumption requires P to be delayed by at least $(w+1)/\epsilon^{i-1}$ packets at e, we arrive at a contradiction Thus, the delay of a packet following a path of length d can be at most $\sum_{i=1}^{d}(w+1)/\epsilon^i$, and therefore SIS must be stable. □

6.3 Instability of FIFO

The FIFO protocol (which is a short form of "first-in-first-out") always gives preference to the packet that was the first to arrive at a node. Ties may be broken arbitrarily. The next result demonstrates that FIFO is not universally stable.

Theorem 6.2 *For $\lambda \geq 0.85$ there is a network and an adversary that causes FIFO to be unstable.*

Proof. The adversary uses the network shown in Figure 1. It is not difficult to check that this network can be embedded in a grid and a 3-dimensional hypercube. Thus, it is possible for our counterexample to happen in important standard networks.

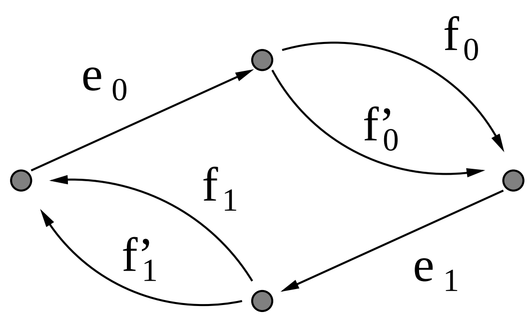

Figure 1: The counterexample for FIFO.

We divide the injection strategy of the adversary into several phases. Our induction hypothesis will be that at the beginning of phase j there are at least $s+j$ packets in the buffer of e_i that intend to cross the edges e_i and f_i, where $i = f \bmod 2$ and s is a sufficiently large constant.

In phase 1 we have to inject packets in such a way that at the end of this phase there are $m = s+2$ packets in the buffer of e_0 that have a remaining path of $(e_0 f_0)$. This can be achieved by attaching a set of m lines of nodes to the starting node of e_0, where line i has a length of i/λ. An injection rate of λ allows m packets to be injected at the endpoints of the lines, one per endpoint, so that all m packets reach e_0 at the same time. This ensures the induction hypothesis for phase 2.

We consider now an arbitrary even phase j. The odd phases work in the same way. We will show that if at the beginning of phase j the buffer of e_0 contains a set M_0 of $m = s+j$ packets with remaining path $(e_0 f_0)$, then at the beginning of phase $j+1$ there will be at least $m+1$ packets in the buffer of e_1 that have a remaining path of $(e_1 f_1)$.

In the following we decribe the injection strategy of the adversary for phase j. To simplify the proof, we will avoid dealing with floors and ceilings. If m is sufficiently large, then their effects can be neglected. Phase j consists of 3 stages.

Stage 1 consists of m steps. During this stage we inject a set M_1 of λm packets with path $(e_0 f_0' e_1 f_1)$. These packets are blocked by the packets in M_0. Furthermore, we inject λm packets with path (f_0). These packets ensure that at the end of stage 1 at least λm packets with remaining path (f_0) are still in the buffer of f_0.

Stage 2 consists of λm steps. During this stage we inject a set M_2 of $\lambda^2 m$ packets with path $(f_0 e_1 f_1)$. These are blocked by the packets that are already in the buffer of f_0. Furthermore, we inject $\lambda^2 m$ packets with path (f_0'). These mix with the packets in M_1. Every $1/\lambda$ steps we have in addition to $1/\lambda$ packets from M_1 that reach the buffer of f_0' one packet with path (f_0'). This has the effect that every $1 + 1/\lambda$ steps $1/\lambda$ packets from M_1 pass f_0', which reduces the number of packets in the buffer of f_0' after stage 2 to

$$\lambda m - \frac{\lambda m}{1 + 1/\lambda} \cdot \frac{1}{\lambda} = \lambda m - \frac{\lambda m}{\lambda + 1} = \frac{\lambda^2 m}{\lambda + 1} \, .$$

Stage 3 consists of $\lambda^2 m$ steps. During this stage, the packets in M_1 and M_2 move forward and mix in the buffer of e_1. Furthermore, $\lambda^3 m$ packets are injected with path $(e_1 f_1)$. Since $\lambda^2 m$ packets traverse e_1, the number of packets in the buffer of e_1 with remaining path $(e_1 f_1)$ after stage 3 is equal to $\lambda^3 m + \lambda^2 m/(\lambda + 1)$.

This ends phase j. Since for $\lambda \geq 0.85$ we have that $\lambda^3 + \lambda^2/(\lambda + 1) > 1$, we arrive at the induction hypothesis for phase $j + 1$. □

6.4 References

- M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 380–389, 1996.