6 Adversarial Queueing Theory II - Networks

Recall the adversarial routing model from the previous section. A scheduling protocol is called *greedy* if whenever there is a packet in some node v that wants to traverse an edge (v, w), v will send a packet along that edge. Apart from ELIS, all queueing disciplines presented in the previous section are greedy. A network G is called *universally stable* if for any greedy scheduling protocol and any (w, λ)-bounded adversary with $\lambda < 1$ the (expected) delay of the packets does not grow unboundedly with time.

In this section we will determine which graphs are universally stable under adversarial injections, and which are not. Most of the results are due to Goel [2]. We distinguish between network topologies that represent directed graphs (i.e. communication is only possible in one direction for every edge, and for every pair of nodes there is at most one edge) and undirected graphs (i.e. every edge can be used in both directions). We assume that the adversary is only allowed to select paths that traverse an edge at most once (no matter which direction is used). That is, every undirected edge can only be used once.

6.1 Universal stability of directed graphs

First, we investigate universal stability for directed graphs. For this the following lemma will be helpful.

Lemma 6.1 *If the directed graphs G_1 and G_2 are universally stable, then so is any directed graph G formed by joining them with edges that go only from G_1 to G_2.*

Proof. Assume that the adversary we are working against has rate $1 - \epsilon$ and burst size w. Since G_1 is universally stable, any packets originating in G_1 get out of G_1 within T_1 time steps, where T_1 is some function of w and ϵ. Some of these packets may then enter G_2. Now consider a time window of size T_2. Any new packets that enter G_2 during this window must have been injected during a time window of size $T_1 + T_2$. The number of paths crossing an edge in G_2 that are introduced during this interval can be at most $(T_1 + T_2 + w)(1 - \epsilon)$. Choose any ϵ' such that $0 < \epsilon' < \epsilon$. Then $T_2(1 - \epsilon') \geq (T_1 + T_2 + w)\lambda$ if and only if $T_2 \geq (T_1 + w)(1 - \epsilon)/(\epsilon - \epsilon')$. This implies that the packets that have to traverse edges in G_2 could have been introduced by a $((T_1 + w)(1 - \epsilon)/(\epsilon - \epsilon'), (1 - \epsilon'))$-bounded adversary. By the definition of universal stability, G_2 is stable against against such an adversary. Hence, the traversal time for a packet as well as the queue lengths are bounded in G. $\sqcap$

Lemma 6.1 implies that all acyclic directed graphs (or short DAGs) are universally stable. Moreover, it implies the following result. (A *strongly connected component* in a directed graph $G = (V, E)$ is a set of nodes $U \subseteq V$ in which there is a directed path from every node to every other node in it.)

Corollary 6.2 *A directed graph is universally stable if and only if all of its strongly connected components are universally stable.*

It remains to show which strongly connected graphs are universally stable. The following result will be useful for this.

Lemma 6.3 ([1]) *The directed cycle is universally stable.*

The next logical question to ask is: Does there exist a strongly connected graph that is more complicated than a cycle and that is still universally stable? The answer is no. Consider the two simple graphs H_1 and H_2 in Figure 1. We will show that neither of these graphs is universally stable. For now, we ignore the fact that the paths specified by the adversary have to be simple. We will come back to it later in the section.

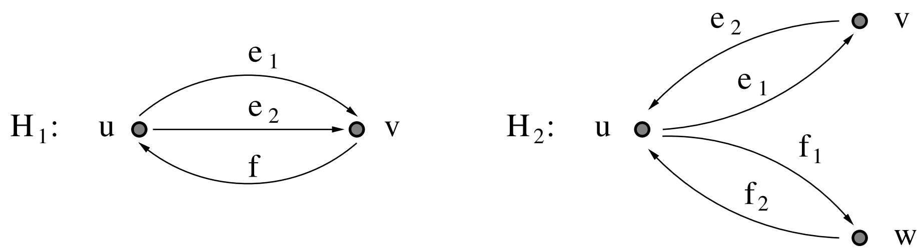

Figure 1: The graphs H_1 and H_2.

Lemma 6.4 *H_1 is not universally stable.*

Proof. To prove instability, the adversary and the greedy protocol are allowed to work in collusion. Suppose that there are s packets waiting to cross edge f, where s is a sufficiently large constant. This can be reached by appending a set L_s of s directed lines of different lengths at u that make sure that packets can be injected at L_s, one per line, so that all of them reach u at the same time. The adversary introduces packets at rate $\lambda < 1$ and operates in four rounds. At the end of the fourth round, there will be more than s packets waiting to cross edge f. Repeating this over and over again will then increase the number of packets in the system to infinity.

The first round lasts for s steps. The adversary introduces λs packets of the form (fe_2) and delays all of these. (Recall that the adversary is also controlling the protocol and therefore can also choose which packets to delay as long as the greedy property is maintained.) At the beginning of the second round, there are λs packets with path (fe_2) waiting at edge f. During the second round, which has a duration of λs, $\lambda^2 s$ packets with path (e_2) and another $\lambda^2 s$ packets with path (fe_1) are introduced. All of these packets are delayed using packets with path (fe_2) left over from round 1. At the beginning of round 3, there are $\lambda^2 s$ packet left for each of the paths (e_2) and (fe_1). The adversary now introduces in $\lambda^2 s$ steps $\lambda^3 s$ packets for each of the paths (e_2) and $(e_1 f)$, delaying them using the packets with paths (e_2) and (fe_1), respectively, from the previous round. At the beginning of the fourth round, there are $\lambda^3 s$ packet left for each of the paths (e_2) and $(e_1 f)$. During the fourth round, which has a duration of $\lambda^3 s$ time steps, the adversary injects $\lambda^4 s$ packets for each of the paths $(e_2 f)$ and (e_1). The new packets with path $(e_2 f)$ are delayed using the packets with path (e_2) left over from the previous round, whereas packets of type (e_1) are allowed to pass through, delaying $\lambda^4 s$ packets with path $(e_1 f)$ from the previous round (the remaining $\lambda^3 s - \lambda^4 s$ packets with path $(e_1 f)$ are allowed to pass through). Notice that this is the only step when the greedy protocol differs from LIS. At the end of the fourth round, there are $\lambda^4 s$ packets left for each of the paths $(e_1 f)$ and $(e_2 f)$. All these packets need to cross edge f.

During the above procedure, the number of packets that need to cross edge f went from s to $2\lambda^4 s$. If $\lambda > 0.5^{1/4} \approx 0.84$, then the number of packets goes up, and the adversary can use this procedure over and over again to make the number of packets in the system unbounded. Thus, the graph resulting from combining L_s and H_1 is not universally stable. Since according to Lemma 6.1 L_s is universally stable, it follows from Corollary 6.2 that H_1 is not universally stable. $\sqcap$

Simple modifications to the above proof result in the following corollary.

Corollary 6.5 *Any graph obtained by replacing the edges in H_1 by disjoint directed paths is not universally stable.*

Lemma 6.6 *H_2 is not universally stable.*

Proof. The proof is similar to that for Lemma 6.4. The adversary again operates in four rounds. Suppose there are s packets waiting to cross edge e_1. In the first round, which has a duration of s steps, the adversary injects λs packets with path $(e_1 e_2 f_1 f_2)$ and delays them. In the second round, which has a duration of λs steps, the adversary injects $\lambda^2 s$ packets for each of the paths (e_2) and (f_2) and delays them. In the third round, which has a duration of $\lambda^2 s$ steps, the adversary injects $\lambda^3 s$ packets for each of the paths $(e_2 e_1)$ and (f_2) and again delays the newly introduced packets. In the fourth round, which has a duration of $\lambda^3 s$ steps, the adversary introduces $\lambda^4 s$ packets for each of the paths (e_2) and $(f_2 e_1)$. During this round, the adversary delays the packets with paths $(e_2 e_1)$ from the previous round and $(f_2 e_1)$ from the current round. Therefore, at the end of the fourth round, there are $2\lambda^4 s$ packets that need to cross edge e_1. $\sqcap$

Corollary 6.7 *Any graph obtained by replacing the edges in H_2 by disjoint directed paths is not universally stable.*

Any strongly connected graph must either be a directed cycle, or it must consist of a directed cycle C and at least one node outside of C that has a directed path p towards C and a directed path q from C, where C, p, and q are edge-disjoint. p and q must be disjoint from C, because they end as soon as they reach one of the nodes of C. Suppose that p and q are not edge-disjoint. Then consider the last node v where they are together before they meet with C. v cannot be a member of C, because p and q have different orientations. Hence, v is outside of C and the rest of the paths p and q are edge-disjoint, as claimed. If p and q meet C at different nodes, the graph would be unstable by Corollary 6.5. If they meet C at the same node, the graph would be unstable by Corollary 6.7. Imposing the restriction that each packet must follow a simple path, the forbidden minors for universally stable directed graphs are given in Figure 2.

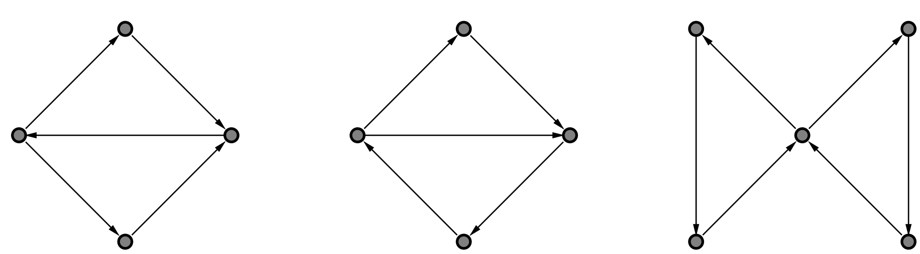

Figure 2: The forbidden minors for a universally stable directed graph.

Theorem 6.8 *A directed graph G is universally stable if and only if it does not contain any of the forbidden minors drawn in Figure 2.*

The above theorem does not indicate whether there exist any natural greedy protocols for which the above characterization holds. However, with the help of simple modifications of the proofs of Lemmas 6.4 and 6.6 one can show the following theorem.

Theorem 6.9 *FIFO is stable for the directed graph G if and only if G is universally stable.*

6.2 Universal stability of undirected graphs

For undirected graphs, Theorem 6.8 translates into an even simpler characterization.

Theorem 6.10 *An undirected graph G with n nodes is universally stable if and only if it has no more than n edges.*

Proof. First, consider an arbitrary undirected tree T. We show that T can be transformed into a DAG T' so that every edge in T occurs only once in T' for each direction and for every simple path in T there is a simple, directed path along the same edge queues in T'.

Select a node in T as the root r and produce two copies of T: T_1 and T_2. In T_1 all edges are oriented towards the root and in T_2 all edges are oriented away from the root. For every node in T_1 there is a directed edge to its copy in T_2. The resulting graph is called T'.

Obviously, T' fulfills the claims above. Since T' is a DAG, it follows from Lemma 6.1 that it is universally stable. Hence, also T must be universally stable.

Next consider an undirected cycle C. When allowing only simple paths on C, then C can be seen as two disconnected directed cycles. Thus, it follows from Lemma 6.3 that C is also universally stable.

Any connected graph with n nodes and $n-1$ edges is a tree, and any graph with n nodes and n edges is a cycle with trees hanging down from it. Using the transformation rules for the cycle and the trees allows us to apply the results about directed graphs to conclude that any graph with n nodes and at most n edges is universally stable.

If a connected graph with n nodes has more than n edges, it must have two cycles. When we bi-direct each cycle, this would result in an unstable minor. $\sqcap$

References

[1] 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)*, pages 380–389, 1996.

[2] A. Goel. Stability of networks and protocols in the adversarial queueing model for packet routing. In *Proc. of the 10th ACM/SIAM Symp. on Discrete Algorithms (SODA)*, pages 911–912, 1999.