Incentive Compatibility and Dynamics of Congestion Control

P. Brighten Godfrey* Michael Schapira† Aviv Zohar‡ Scott Shenker§

ABSTRACT

This paper studies under what conditions congestion control schemes can be both *efficient*, so that capacity is not wasted, and *incentive compatible*, so that each participant can maximize its utility by following the prescribed protocol. We show that both conditions can be achieved if routers run strict priority queueing (SPQ) or weighted fair queueing (WFQ) and end-hosts run any of a family of protocols which we call Probing Increase Educated Decrease (PIED). A natural question is whether incentive compatibility and efficiency are possible while avoiding the per-flow processing of WFQ. We partially address that question in the negative by showing that any policy satisfying a certain "locality" condition *cannot* guarantee both properties.

Our results also have implication for *convergence* to some steady-state throughput for the flows. Even when senders transmit at a fixed rate (as in a UDP flow which does not react to congestion), feedback effects among the routers can result in complex dynamics which do not appear in the simple topologies studied in past work. Interestingly, we find that the same locality condition which precludes incentive compatibility and efficiency also implies qualitatively different *convergence* properties, compared with SPQ and WFQ which are guaranteed to converge to a fixed point.

1. INTRODUCTION

Congestion control is a crucial task in communication networks which occurs through a combination of mechanisms on *end-hosts* through protocols such as TCP; and on *routers and switches* through use of queue management schemes such as WEIGHTED FAIR QUEUEING (WFQ) [2, 23] or FIFO queueing with DROP TAIL. In the schemes used in practice, most commonly TCP and FIFO queueing with DROP TAIL, it is well known that senders can improve their throughput by misbehaving—for example, increasing their transmission rate beyond that of TCP, thus inducing other senders to back off. Thus, this suite of protocols lacks *incentive compatibility*: the property that each participant can maximize its utility by following the prescribed protocol. In a network such as the Internet, in which senders are controlled by different entities which can easily deviate from the protocol with limited repercussions, incentive compatibility is a useful way to obtain predictable performance. While other competitive aspects of congestion control such as fairness [2, 23] and the price of anarchy [17] have been studied extensively, little attention has been paid to incentive compatibility in this context.

Incentive compatibility depends both on the end-host protocol and the queueing scheme used by routers. FIFO queueing with DROP TAIL, which treats all packets equally regardless of the rate at which the sender is transmitting, is incentive compatible only with the end-host protocol that sends packets as quickly as possible. This behavior causes packet loss and thus *inefficiency* in the network.

This paper contains contributions in two areas. First, we study under what conditions both *incentive compatibility and efficiency* can be obtained in arbitrary networks. Second, we study under what conditions flow rates *converge* to a fixed point in arbitrary networks, which is both a step in our incentive compatibility analysis and is of independent interest. We describe our results in these two areas next.

Incentive compatibility. We present a family of end-host congestion control protocols called PROBING INCREASE EDUCATED DECREASE (PIED), in which the source gradually increases its transmission rate until it encounters packet loss, at which point it decreases its sending rate to the throughput that the receiver observed, unlike TCP, which backs off more dramatically. We show that if each end-host runs a PIED protocol

*Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA. *pbg@illinois.edu*

†Department of Computer Science, Yale University, CT, USA, and Computer Science Division, University of California at Berkeley, CA, USA. Supported by NSF grant 0331548. *michael.schapira@yale.edu*

‡School of Engineering and Computer Science, The Hebrew University of Jerusalem, Jerusalem, Israel and Microsoft Israel R&D Center, Herzlia, Israel. *avivz@cs.huji.ac.il*

§Computer Science Divison, University of California at Berkeley, and International Computer Science Institute, Berkeley, CA, USA. *avivz@cs.huji.ac.il*

(not necessarily the same one), and the routers use a queueing policy like WFQ or Strict Priority Queueing (SPQ) (with coordinated weights or priorities), then the network converges to a fixed point. Moreover, this fixed point is efficient, in the sense that there is no packet loss and no needlessly unused link capacity. We show that this convergence process is also incentive compatible and collusion-proof: assuming that end-hosts care about their throughput, no sender or coalition of conspiring senders can obtain a better throughput by running a protocol other than PIED. These results follow intuitively from the isolation between flows provided by WFQ and SPQ.

A natural question is whether incentive compatibility and efficiency are possible while avoiding the per-flow processing of WFQ and SPQ. Consider, for example, the following queueing policy, which we call *local weighted fair queueing (LWFQ)*: at each router, packets are grouped into "superflows" such that all packets in each superflow have the same previous and next hops; then, WFQ is run among these superflows. LWFQ clearly doesn't have the same fairness properties as WFQ, but does it provide enough isolation to enable incentive compatibility?

Unfortunately, this is not the case. In fact, we show that *any* queueing scheme which is *local* — in the sense that treatment of packets at each router depends only on the portion of the packet's path near the router — cannot be both incentive compatible and efficient. The intuition is that any local scheme can be made to behave like Drop Tail on some topologies. This indicates that obtaining incentive compatibility requires examination either of remote portions of the packet's path or of other information carried by the packet, as in CSFQ [25].

Convergence. A key step in the above analysis, and a fundamental property of independent interest, is *convergence* of the network to an equilibrium, in terms of the rates that packets are sent and received on end-to-end flows. A long line of research has characterized the dynamic behavior of various combinations of end-host protocols and router queueing policies. The large majority of past work focuses on the case of a single congested router or on simple networks such as a series of several congested routers [8]. In this paper, we study the general case in which the network is arbitrary and there may be *multiple points of congestion.*

Suppose each source in a network sends at a fixed rate, such as a UDP flow which does not react to congestion. One might expect that because the flow is *sent* at a constant rate, it will therefore be *received* at a constant rate (albeit at a lower rate than the flow was sent, if it encounters congestion). This is true for simple networks such as a single router or series of several routers. However, we show that complex feedback behavior among routers, absent in the well-studied simpler topologies, can arise in the general case. Our results for queueing policies give a spectrum of convergence behavior:

- If routers use WFQ or SPQ with weights or priorities that are not coordinated across routers (as may occur, for example, due to a configuration error or simply different policies in different autonomous systems), flows' throughputs may permanently oscillate. Throughputs may also converge to one of multiple different equilibria depending on initial conditions and timing.
- When routers use WFQ or SPQ and each flow has the same priority or weight on all the links it traverses, we show that the flows' throughputs are guaranteed to converge to a single fixed point in finite time.
- If routers use Drop Tail or any other local queueing policy (according to the above definition), only asymptotic worst-case convergence is possible, unlike the exact convergence of WFQ and SPQ. We show that a fixed point always exists; and the network will converge asymptotically to this fixed point if there is only a single "cycle of dependencies" (to be defined precisely later). We leave open the question of whether convergence is guaranteed in general.

Interestingly, our sufficient condition for exact convergence (i.e., using WFQ or SPQ) is also sufficient for incentive compatibility and efficiency; and our necessary condition for exact convergence (i.e., non-locality) is also necessary for incentive compatibility and efficiency. Note, however, that there is a large gap between these necessary and sufficient conditions.

Our results use a fluid-flow model for analysis. Although the model is simple, it allows us to expose and analyze basic incentive and feedback effects. Moreover, we do not claim that the convergence problems that we demonstrate are likely to be common in real-world networks. Instead, we believe the importance of this work is analogous to the fact that BGP routes on the Internet may oscillate [26]: given the importance of incentives, efficiency, and convergence, it is desirable to know how a network designer can guarantee these properties.

Organization of the paper. The remainder of the paper proceeds as follows. We introduce our model of the network in Sec. 2. We analyze convergence with fixed-rate senders in Sec. 3, and incentive compatibility and efficiency in Sec. 4. We discuss related work in Sec. 5 and conclude in Sec. 6.

2. MODEL

2.1 The Network Model

We represent the communication network by a directed graph $G = (V, E)$, where the set of vertices V

represents routers and end-hosts, and the set of edges E represents data links. Each directed edge $e \in E$ has a *capacity* $c_e > 0$, that represents the link's bandwidth. Each end-to-end connection (flow) in the network is defined by a pair of *source-destination* vertices (the communicating hosts), and a path in the network through which data flows between these end-hosts. The path that connects two end-hosts is determined by underlying routing protocols, and is considered fixed for the purposes of this model. The transmission rate of each source vertex i is bounded from above by some maximum possible influx $\overline{f_i} > 0$.

At each point in time, each flow is transmitted with some (real-valued) rate across each link. Our goal is to determine, given arbitrary initial rates, whether the flow rates will converge, and if so, to what values. These congestion control dynamics are determined by two algorithms: the routers' *queue management policies*, which dictate how routers discard excess traffic when links' capacities are exceeded, and the *congestion control protocol* executed by the end-hosts, which dictates the way in which source nodes adjust their transmission rates. The timing of when these processes react depends on non-deterministic delays due to processing, link latency, end-to-end packet acknowledgements, and so on. To model this uncertainty, our model assumes essentially arbitrary adversarially-controlled timings of the activations of these processes, subject to the assumption that the routers' processes react on faster timescales than end-hosts.

More formally, our model alternates between end-host events and network convergence. We consider an infinite sequence of discrete time steps $t = 1, 2, \ldots$ at which end-host reaction occurs. These discrete moments denote only ordering between events, rather than absolute real-time values. In each time step, some subset of the end-to-end connections is *activated* and adjust their transmission rates subject to their congestion control protocols. Immediately following each end-host activation at time t, the routers' reaction occurs at discrete timesteps $t + \varepsilon, t + 2\varepsilon, \ldots$ for some negligible ε, until the routers have converged. Similar to the activation of end-hosts, at each time $t + i\varepsilon$ some subset of the routers are activated and adjust their flow rates subject to the queue management policy. We assume that the timing of activations does not permanently starve any end-host or router of activations, but is otherwise arbitrary.

Because this model assumes convergence of the routers' flow rates given fixed sender transmission rates, our analysis will begin in Sec. 3 with a characterization of when such convergence is guaranteed.

2.2 Queue Management Policies

Routers' queue management policies specify how a link's capacity is shared between the flows that reach that link. For every edge $e \in E$, we denote the end-to-end connections whose paths go through e by $K(e)$. For every connection i, we denote i's flow at a node u on its path by $f_i(u)$. We then define a queue management policy as follows.

DEFINITION 1. *Let $e = (u, v) \in E$ and let $1, \ldots, k$ be the connections in $K(e)$. A queue management policy for e is a function*

$$Q_e : (\mathbb{R}_+)^k \rightarrow (\mathbb{R}_+)^k$$

that maps every k-tuple of incoming flows $f_1(u), \ldots, f_k(u))$ to a k-tuple of outgoing flows, or "capacity shares", $(\tilde{f}_1(v), \ldots, \tilde{f}_k(v))$, such that:

- $\forall i \in \{1, \ldots, k\}$ $f_i(u) \geq \tilde{f}_i(v)$ *(a connection's flow leaving the edge cannot be bigger than that connection's flow entering the edge).*
- $\sum_{i=1}^{k} \tilde{f}_i(v) \leq c_e$ *(the sum of connections' flows leaving the edge cannot exceed the edge capacity).*

We next define in the context of our model the queue management policies that we will analyze in the remainder of the paper.

Strict Priority Queueing (SPQ). SPQ assumes that types of traffic can be differentiated and assigned to separate FIFO queues with higher priority queues always processed to completion before lower priority queues are considered. More formally:

DEFINITION 2. *An edge $e = (u, v)$ has a* STRICT PRIORITY QUEUEING *policy if it allocates capacity in the following manner: There is some fixed order over the connections in $K(e)$, $1, \ldots, k$.*

- *If $f_1(u) \geq c_e$ then 1 is granted the entire capacity of the edge, c_e.*
- *Otherwise, connection 1 gets its full demand $f_1(u)$, and the remaining capacity $c_e - f_1(u)$ is allocated to the remaining connections $2, \ldots, k$ by recursively applying the same procedure.*

We say that the priorities of connections are *coordinated* across links if, whenever connection A is prioritized over another connection B in some link, A is prioritized over B at all of their shared links.

Weighted Fair Queueing (WFQ). [23] WFQ enforces weighted max-min fairness among flows at a router. In a fluid model, each flow with weight w at a link where the flows' weights sum to W is allocated a fraction w/W of the link's capacity; any spare capacity (resulting from flows whose incoming rate is less than this fair share) is then recursively allocated among the flows whose incoming rates are more than the fair share.

Definition 3. *An edge $e = (u,v)$ has a* Weighted Fair Queueing *policy if it allocates capacity in the following manner: There are nonnegative real numbers ("weights") $w_1, ..., w_k$ associated with the connections $1, ..., k$ in $K(e)$.*

- *If, for each $i \in K(e)$, $f_i(u) > \frac{w_i \cdot c_e}{\Sigma_{r \in K(e)} w_r}$, assign each connection i a capacity share of $\frac{w_i \cdot c_e}{\Sigma_{r \in K(e)} w_r}$.*
- *If, for some $j \in K(e)$, $f_j(u) \le \frac{w_j \cdot c_e}{\Sigma_{r \in K(e)} w_r}$, grant connection j its full demand $f_j(u)$ and repeat the same procedure recursively for the remaining capacity of $c_e - f_j(u)$ and the remaining $k-1$ connections.*

We say that connections' weights are *coordinated* across links if every connection has the same weight on each edge on its path.

Fair Queueing (FQ). **[2]** FQ is the special case of Weighted Fair Queueing in which the weight of each connection is 1.

Drop Tail (DT). DT is perhaps the simplest and most common queue management algorithm: packets are placed in a single FIFO queue until it is full, whereupon any newly arriving packets (the tail) are dropped. Since each packet is treated identically regardless of the incoming flow rate, a connection's capacity-share is proportional to its incoming flow:

Definition 4. *The* Drop Tail *queueing policy at edge $e = (u,v)$ allocates to each connection $i \in K(e)$ the capacity share $\frac{f_i(u)}{\Sigma_{j \in K(e)} f_j(u)}$.*

Local Queue Management Policies. General queueing policies such as Weighted Fair Queueing and Strict Priority Queueing may require identification of each end-to-end flow. We now introduce a property called *locality* which is a class of policies that avoids this per-flow processing.

Informally, we call a policy *local* when the handling of a connection depends only on its neighborhood at a router, *i.e.*, the previous and next hops. In other words, at each router, flows are grouped into "superflows" according to (input port, output port) pair. An arbitrary policy manages queueing among superflows, and each individual connection receives a share of its superflow's bandwidth which is proportional to the connection's share at the input port.

Definition 5. *An edge $e = (u,v)$ has a local queue management policy if the following holds: Let $e_1, ..., e_t$ be u's incoming edges in G. There is a function $g : (\mathbb{R}_+)^t \rightarrow (\mathbb{R}_+)^t$, such that for every e_j, and every connection $i \in K(e)$ whose route goes through e_j, i's capacity-share on e is $\frac{f_i(u)}{F_{e_j}} \cdot g_j(F_{e_1}, \ldots, F_{e_t})$, where F_{e_r} denotes the total sum of the flows at u of the connections in $K(e)$ whose paths traverse the edge e_r.*

The simplest local queueing scheme is Drop Tail. We also define Local Weighted Fair Queueing (LWFQ), which is a local analogue of Weighted Fair Queueing: Let $e = (u,v)$, and let $e_1, ..., e_t$ be u's incoming edges. The packets of the flows in $K(e)$ traversing each incoming edge e_i are grouped into a "superflow". Then, WFQ is run among these superflows. Each connection gets its proportional share in the capacity share allocated to the superflow to which it belongs.

3. CONVERGENCE WITH FIXED RATE SENDERS

The main focus of this paper is to characterize incentive compatibility and efficiency, which involves *end-host dynamics* (see Sec. 4). However, to address this question, we must first understand the network's behavior in the seemingly simple scenario that all sources are transmitting at a *fixed* rate. One might expect that this case should always converge to fixed throughput rates. In Section 3.1 we give several simple examples that show that even when senders send at constant rates, the system can have complex dynamics, including multiple stable states and persistent oscillation. Hence, with a poor choice of router queueing policies, oscillations can be inherent to the network, rather than being caused by the end-hosts. In Section 3.2 we give strong guarantees for WFQ and SPQ with coordinated weights and priorities: these policies have a single fixed point to which they converge in finite time. In Section 3.3 we show that local queueing policies including Drop Tail have qualitiatively different convergence properties, in that they can only guarantee asymptotic convergence. We also show that DT does guarantee the existence of at least one fixed point, and that in the special case that there is a single cycle of dependencies, it will (asymptotically) converge.

3.1 Convergence is Nontrivial

Example 1: Consider the network of Figure 1. There are two connections C^1, C^2. All edge capacities equal 1. The routers use SPQ with uncoordinated priorities such that each edge prioritizes packets that have traveled a longer distance before arriving at that edge. For example, consider the edge $(1,2)$, used by both connections. Packets from C^2 that reach this edge have traveled a longer distance at that point, so they are always preferred over packets from C^1. Similarly, at edge $(3,4)$ the connection C^1 is preferred over C^2.

Consider the case that C^1 and C^2 start transmitting simultaneously at a fixed rate of 1. At first, all of C^1's traffic would go through the edge $(1,2)$, and, similarly, all of C^2's traffic would go through the edge $(3,4)$. Be-

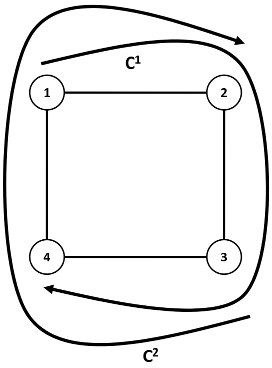

Figure 1: A network in which Strict Priority Queueing **with uncoordinated priorities has infinitely many stable states, but may also oscillate indefinitely.**

cause each of the edges $(2,3)$ and $(4,1)$ is used by exactly one connection (no competition), this means that next all of C^1's flow would reach vertex 3, and all of C^2's flow would reach vertex 1. However, observe that because C^1 is preferred at $(3,4)$ and C^2 is preferred at $(1,2)$, this implies that all traffic of the less preferred connection on these edges will now be discarded. Hence, after awhile C^1's traffic will not reach vertex 3 and C^2's traffic will not reach vertex 1, thus re-enabling C^1 and C^2 to get the full capacities of $(1,2)$ and $(3,4)$, respectively. Observe that this brings us back to the initial state, and so this process can go on indefinitely, never reaching a stable state.

We observe that in this small network there are infinitely many stable states: Consider a flow pattern in which C^1 is assigned a capacity share of $p \in [0,1]$ on each edge on its path, and C^2 is assigned a capacity share of $(1-p)$ on each edge on its path. This flow pattern is stable.

Example 2: The example of Figure 2 shows that oscillations can occur even if a unique stable state exists in the network. There are three connections C^1, C^2, C^3. All edge capacities equal 1. Once again, the routers are all using SPQ and each edge prioritizes packets that have traveled a longer distance before arriving at that edge. The flow pattern in which every connection gets a capacity share of $\frac{1}{2}$ on each of the edges on its path is stable. It is easy to show that this is, in fact, the *unique* stable state in this network.

Now, consider the case in which C^1 and C^2 start transmitting simultaneously, at a rate of 1. Since C^1 and C^2 share the edge $(1,2)$ and since C^1 is preferred at that edge, it is given the full capacity 1, while all packets that belong to C^2 are discarded. Now, assume that, at this point, C^3 also starts sending 1 unit of flow. Since C^3 is preferred at edge $(3,1)$ and has no competition over the available capacity in $(2,3)$, it is given precedence and suppresses the traffic of C^1. This in turn allows C^2 to send flow freely and suppress C^3, and so on. This can go on indefinitely, never reaching a stable outcome.

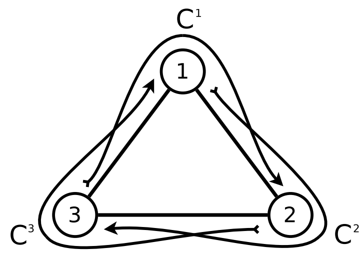

Figure 2: A network in which Strict Priority Queueing **with uncoordinated priorities has a single stable state, but may oscillate indefinitely.**

Both of the examples above can also be shown to hold when the routers run WFQ with uncoordinated weights. This is because WFQ can closely approximate SPQ by assigning a very big relative weight to more preferred connections.

3.2 Coordinated WFQ and SPQ

The examples in Figure 1 and 2 show that oscillations are possible when SPQ and WFQ use uncoordinated priorities or weights. We now prove that for SPQ and WFQ with *coordinated* parameters, convergence is guaranteed (for any network topology). The proofs of both results have a similar structure: We show that coordinated priorities/weights imply an "isolation" between connections. This enables the application of a recursive proof technique in which connections/edges are gradually removed from consideration.

Theorem 3.1. *For any network topology, if all connections transmit at a fixed rate, all routers use SPQ, and connections' priorities are coordinated across links, then convergence to a stable flow pattern is guaranteed.*

Proof. Consider the connection i that has the highest priority on all links it traverses (since priorities are coordinated across links such a connection is bound to exist). Let e be the link on i's route with the smallest capacity. There are two possible cases:

Case I: If i's (fixed) transmission rate (initial influx) $\overline{f_i}$ is at most c_e then i's throughput is guaranteed to be $\overline{f_i}$. In this case, Strict Priority Queueing will allocate i a capacity share of $\overline{f_i}$ on each link it traverses, and share unused capacity between the other flows in a recursive manner. Hence, after some time goes by, the network is effectively equivalent, in terms of convergence, to a network in which connection i is removed and the capacity of each link on its route is updated accordingly.

Case II: If i's (fixed) transmission rate (initial influx) $\overline{f_i}$ is greater than c_e then observe that i will be assigned all of e's capacity, and that, after awhile it will be assigned a capacity share of exactly c_e on every edge that comes after e on i's route. Hence, after some time goes by, the network is effectively equivalent, in terms of convergence, to a network in which connection i's route is shortened and ends just before e, edge e is removed, and the capacity of all links following e on i's original route are updated accordingly.

For both of the cases described above we show that, after some time goes by, the network can be reduced to a smaller network (by effectively removing an edge or a connection). Every such reduction fixes the flow of a connection across at least a single link. The same line of argument can be recursively applied until all connections' flows remain fixed on all edges. □

A proof similar to that of Theorem 3.1 enables us to show a similar result for WFQ:

THEOREM 3.2. *For any network topology, if all connections transmit at a fixed rate, all routers use WFQ, and connections' weights are coordinated across links, then convergence to a stable state is guaranteed.*

PROOF. Let e be the link in G for which the expression $\frac{c_e}{\Sigma_{r\in K(e)}\ w_r}$ is minimized. Let α denote this value. We handle two cases:

Case I: If, for some $i \in K(e)$, i's (fixed) transmission rate (initial influx) $\overline{f_i}$ is at most $w_i \cdot \alpha$, then observe that i's throughput is guaranteed to be $\overline{f_i}$. This is because on any other link the value of α would be greater, and so the capacity allocated to i is greater. In this case, WEIGHTED FAIR QUEUEING will allocate i a capacity share of $\overline{f_i}$ on each link it traverses, and share unused capacity between the other flows in a recursive manner. Hence, as in the proof of Theorem 3.1, the network is effectively equivalent, in terms of convergence, to a network in which connection i is removed and the capacity of each link on its route is updated accordingly.

Case II: If, for every $i \in K(e)$, i's (fixed) transmission rate (initial influx) $\overline{f_i}$ is greater than $w_i \cdot \alpha$ then observe that each such connection i will be eventually be assigned a capacity share of $w_i \cdot \alpha$ by link e and a capacity share that is no smaller by any other link on its path (by the definition of α). Therefore, after awhile each $i \in K(e)$ will be assigned a capacity share of exactly $w_i \cdot \alpha$ on every edge that comes after e on i's route. Hence, after some time goes by, the network is effectively equivalent, in terms of convergence, to a network in which each such connection i's route is shortened and ends just before e, the capacity of all links following e on each i's original route (including e) are updated accordingly, and e is removed from the network.

For both of the cases described above we show that, after some time goes by, the network can be reduced to a smaller network (by effectively removing an edge or a flow). Every such reduction fixes the throughput of a connection across at least a single link. This argument can be recursively applied until all connections' flows on all edges remain fixed. □

3.3 Drop Tail and Local Queue Management

3.3.1 Convergence Properties

While SPQ and WFQ with coordinated weights converge exactly, the following example shows that DROP TAIL may only approach its fixed point asymptotically. While this may be acceptible in practice, it does demonstrate a qualitative difference in the behavior of DT vs. SPQ and WFQ. We also show that any local queueing policy which is efficient (in the sense that it uses all available bandwidth) is as bad as DT in this regard.

Consider the network in Figure 3. There are two connections – C^1 and C^2. C^1's route (leading from s_1 to t_1) is described by the dashed arrows and C^2's route (leading from s_2 to t_2 is described by the non-dashed arrows. All edge capacities, as well as both connections' fixed transmission rates, equal 1. All edges use DROP TAIL. Along each dashed/non-dashed line is a variable indicating the appropriate connection's capacity share on that edge. For example, C^1's capacity share on the link (v_1, v_2) is denoted by a. Observe that C^1's capacity share on the link (v_2, v_3) must also be a as C^1 is the only connection using this edge. Also observe that because of symmetry, if C^1's share on (v_1, v_2) is a then so must C^2's share on (v_3, v_4) be. Thus, two variables, a and b, are sufficient to denote all capacity shares of both connections on different links.

Now, consider the link (v_1, v_2). Observe that DROP TAIL queueing implies that C^1's capacity share on this link, a, must equal $\frac{1}{1+a}$. By solving the equation $a = \frac{1}{1+a}$ we get that the only (nonnegative) solution is that $a = -1 + \frac{\sqrt{5}}{2}$. We argue that the irrationality of a implies that C^1's capacity share on (v_1, v_2) will never converge to a fixed number. To see why this is true, simply observe that since the transmission rates of both connections are rational, and every router updates its flow rates using rational operations, it can never happen that a connection gets an irrational capacity share on any link. Hence, C^1 will never get *exactly* a share of $-1 + \frac{\sqrt{5}}{2}$ on (v_1, v_2).

We now show that this example can easily be made to hold for any *efficient* local queue management policy, where by saying that a queueing policy is efficient

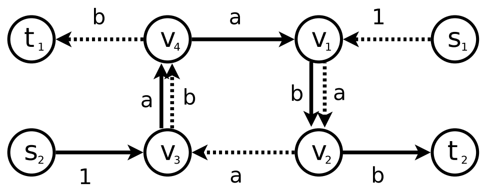

Figure 3: A network in which Drop Tail **converges only asymptotically.**

we mean that there is no needlessly unused link capacity (as is the case with Strict Priority Queueing, Weighted Fair Queueing, and Drop Tail). Replace each directed edge e in Fig. 3 by two consecutive edges, e_1 and e_2, such that e_1 has a very big capacity, and e_2 is identical to e. Observe that this construction guarantees that all traffic reaching e_1 will also reach e_2 (as e_1's queue management policy is efficient), and that e_2 discards of excess traffic exactly as in Drop Tail (by the efficiency and locality of and e_2's queueing policy, and the fact that it only has one incoming edge). Applying the above arguments to this new network leads to the same conclusions as before.

3.3.2 Aymptotic Convergence of Drop Tail

The example in Figure 3, while showing that for local and efficient local queueing polices exact worst-case convergence is not possible, does not rule out the possibility that convergence with DT may be achievable *in the limit.* Here, we explore DT's convergence. We start by proving the following theorem:

Theorem 3.3. *For any network, if all routers use the DT policy, then there exists a stable flow pattern.*

Proof. We prove the theorem using a fixed point argument. We begin by defining the following function $F(\vec{f}) = F(f^1_{e_1}, \ldots, f^m_{e_n})$ for each network configuration. The parameters of this function are the values of flows of all connections at each edge in the network. The function's range is the same as its domain. $F(\vec{f})$ is defined as follows: given a vector of connections' flows per edge, F outputs, for every edge, the flows of the connections on that edge that result from updating that edge's capacity allocation according to Drop Tail., *i.e.*, the amount of flow that each connection would have on that edge after that edge alone is updated from the state $\vec{f}$. Observe that because Drop Tail is a continuous function, so is F, and that the domain of F (which is the same as its range) is a cartesian product of simplexes (the allowed values for flows on the same edge must sum to a number that is lower than its cost – if there are k flows then this is simply the $k+1$-dimensional simplex).

Therefore, we have shown that F is a continuous function from a compact closed set to itself, and must therefore have a fixed-point $\vec{f^*}$. Note that this fixed point is also a stable state, since the flows on each edge will not change their values when that edge alone is updated (by the definition of F). $\square$

It is still unclear whether there is always a *unique* stable flow pattern in networks with Drop Tail queue management policies. In addition, we do not know whether from *any* initial flow configuration the network would eventually converge (in the limit) to a stable state under any timing. These are left as intriguing open questions.

We take a first step towards answering these questions, by showing that convergence to a unique stable state is guaranteed for topologies with at most a single feedback cycle. We conjecture that this result actually holds for all network topologies.

To better understand the dependencies between connections we define the "*dependency graph*": The dependency graph is a directed graph $G_d = (V_d, E_d)$ such that:

- Each vertex in V_d, f^i_e, represents the flow of connection i on link e in our network (where e is an edge on connection i's route).
- There is a directed edge $e = (f^i_{e_1}, f^j_{e_2})$ in E_d iff e_1 and e_2 are two *consecutive* edges on i's route (where e_2 comes directly after e_1), and j's route goes through e_2. Intuitively, an edge between two vertices in G_d implies that the flow of some connection directly affects the flow of another connection.

We prove the following theorem, that applies to networks like the one depicted in Figure 3:

Theorem 3.4. *If the dependencies graph G_d contains at most a single directed cycle, all sources transmit at fixed rates, and all routers use* Drop Tail *queueing, then there is a unique stable flow pattern, and the flow rates on each link approximate it arbitrarily closely as time advances.*

The proof of Theorem 3.4 appears in Appendix A.

4. INCENTIVE COMPATIBILITY AND EFFICIENCY

In Section 3 we dealt with the case that sources transmit data at a constant rate. We now move to the case that sources adjust their transmission rates dynamically. If sources are *selfish* in choosing these rates, when can we guarantee that sources have incentive to follow the protocol, the network would still converge to a fixed point, and this point will be efficient?

4.1 Efficiency and Incentive-Compatibility vs. Local Queueing Policies

Consider the case of two connections, C^1 and C^2, that send traffic over a *single* edge e that uses DROP TAIL queueing. The maximal transmission rate of both C^1 and C^2 is 2, and the capacity of e is 1. Observe that, no matter what congestion control protocol C^2 is using to determine how to adjust its transmission rate, C^1 can always increase its throughput by transmitting at its maximal rate, if it is not already doing so (because DROP TAIL allocates C^1 its proportional share). Therefore, the only incentive-compatible end-host protocol in this case is the protocol that instructs end-hosts to always send packets as quickly as possible. This will, of course, result in packet loss.

We now re-use the construction presented in Sec. 3.3 to show that incentive-compatibility can lead to packet loss for all local and *efficient* queueing policies: Substitute the edge in this example by two consecutive edges, e_1 and e_2, such that e_1 has enormous capacity, and e_2 is identical to e. Let us assume that both e_1's and e_2's capacity is allocated according to some local and efficient queueing policy. Observe that, by the efficiency of the queueing policy of e_1, we have that all traffic that reaches e_1 also reaches e_2. Because e_2 only has one incoming link (namely, e_1), the fact that its queueing policy is local and efficient implies that the allocation of its capacity between C^1 and C^2 is the same as in DROP TAIL. This, in turn, implies that each connection is always rationally motivated to transmit at its maximal rate, leading to packet loss.

In fact, observe that, using the exact same construction, it is possible to show that the same result holds for local routing policies that only utilize a constant fraction of the link in case of congestion (e.g., RANDOM EARLY DETECTION). It is also easy to extend the result to the case that the queueing policy is a function of a larger neighborhood, such as the portion of the route of each connection which is within $O(1)$ hops.

4.2 WFQ and SPQ are Incentive Compatible

4.2.1 Probing Increase Educated Decrease

We now present a family of congestion control protocols called "PROBING INCREASE EDUCATED DECREASE" (PIED). PROBING INCREASE EDUCATED DECREASE protocols are motivated by examples like the following one.

Consider the network graph in Figure 4. There are two connections, C^1 and C^2, whose paths share a single edge. All edges use FAIR QUEUEING. The connections attempt to send 4 units of traffic each. The flow of C^1 is immediately reduced to 3 units at the edge (s_1, v_1) while C^2 manages to transmit the full 4 units to vertex v_1. At this point, both connections receive an equal share

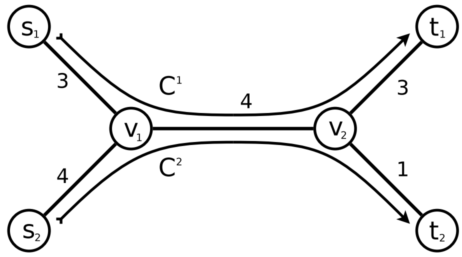

Figure 4: A graph in which a flow wastes network resources needlessly

of 2 flow units along (v_1, v_2). Connection C^1 therefore has 2 units of flow arriving at the target node t_1, while connection C^2's flow is reduced further to only 1 unit that arrives at t_2. Notice however, that if C^2 only sends 1 unit of flow then this unit reaches t_2, and moreover, C^1 then has 3 units of flow arriving at the destination. In this case, the overall network performance is better as the two connections manage to get through 4 units of flow rather than 3.

To avoid such undesirable scenarios, PROBING INCREASE EDUCATED DECREASE protocols are designed to ensure that a connection's transmission rate will (eventually) match its throughput. This is achieved via the following simple rate-adjustment rule: Each source gradually increases its transmission rate until it encounters packet loss, at which point it decreases its sending rate to the throughput that the receiver observed. Different PROBING INCREASE EDUCATED DECREASE protocols differ in the increase factors of the senders, *i.e.*, a sender may additively increase its transmission rate by some constant $\epsilon > 0$, or multiply its transmission rate by some factor as long as it does not encounter packet loss. All of our results hold for all members of the family of PROBING INCREASE EDUCATED DECREASE protocols, and even for cases in which different end-hosts use different protocols within this natural family of protocols.

4.2.2 Convergence of PIED

We prove the following theorems for WEIGHTED FAIR QUEUEING and STRICT PRIORITY QUEUEING with coordinated priorities/weights:

THEOREM 4.1. *For any network topology, and any initial transmission rates, if all connections run* PROBING INCREASE EDUCATED DECREASE *protocols, and all routers use* STRICT PRIORITY QUEUEING *with coordinated priorities, then the congestion control dynamics converge to an equilibrium point in which all connections transmit at a constant rate. Moreover, this fixed point is efficient (there is no packet loss and no needlessly unused link capacity).*

THEOREM 4.2. *For any network topology, and any initial transmission rates, if all connections run* PROBING INCREASE EDUCATED DECREASE *protocols, and all routers use* WEIGHTED FAIR QUEUEING*, then the congestion control dynamics converge to an equilibrium point in which all connections transmit at a constant rate. Moreover, this fixed point is efficient (there is no packet loss and no needlessly unused link capacity).*

The proofs of these theorems are similar in spirit to the proofs of Theorems 3.1 and 3.2 (respectively), and are omitted due to space constraints (available upon request).

4.2.3 Incentive Compatibility

We have shown that for STRICT PRIORITY QUEUEING and WEIGHTED FAIR QUEUEING with coordinated priorities/weights, PROBING INCREASE EDUCATED DECREASE protocols are guaranteed to converge to an efficient state in which all connections transmit at a fixed rate. We now show that these protocols are also *incentive compatible.* That is, we consider the case that end hosts care about their throughputs. *Incentive-compatibility* is a property of PROBING INCREASE EDUCATED DECREASE means that no end-host can obtain a better throughput by running a protocol other than PROBING INCREASE EDUCATED DECREASE. A stronger requirement is that of *collusion proofness*, which means that even a coalition of conspiring end-hosts cannot *strictly* better the throughput of *every* member by deviating from PROBING INCREASE EDUCATED DECREASE.

THEOREM 4.3. *For any network topology, if all routers use* STRICT PRIORITY QUEUEING *with coordinated priorities, then* PROBING INCREASE EDUCATED DECREASE *is collusion-proof.*

PROOF. By contradiction, assume that there is a coalition T (of size at least 1) of manipulative connections that can deviate from PROBING INCREASE EDUCATED DECREASE and better the throughput of each member in the coalition.

Consider the connection i that has the highest priority on all links it traverses. We shall now show that $i \notin T$. Let e be the link on i's route with the smallest capacity. Observe that if i's maximal transmission rate $\overline{f_i}$ is at most c_e then, if all connections execute PROBING INCREASE EDUCATED DECREASE, i's throughput is guaranteed to be $\overline{f_i}$ from some moment in time onwards. Hence, i cannot be in T because it is impossible for i to better its throughput. We are left with case that $\overline{f_i} > c_e$. Observe that, in this case, if all connections execute PROBING INCREASE EDUCATED DECREASE, eventually i's throughput will be c_e. However, in this case, too, it is impossible for i to improve its throughput because it cannot possibly get a higher throughput than e's capacity, and so $i \notin T$.

Now, consider the outcome reached after the deviation from PROBING INCREASE EDUCATED DECREASE. Observe that if $i \notin T$, then i must be running PROBING INCREASE EDUCATED DECREASE, and so, after some time goes by, i is guaranteed to obtain a throughput of exactly $min\{\overline{f_i}, c_e\}$, which is the maximum feasible throughput for i no matter what the other connections do. PROBING INCREASE EDUCATED DECREASE dictates that, from that moment onwards, i's transmission rate will exactly equal its throughput, and so the network is effectively equivalent to a network from which i is removed and all edge capacities along its route are updated accordingly. We can now apply the same arguments to show that the connection with the highest priority in the resulting network (that is, the connection with the second-highest priority in the original network) cannot be in T.

A repeated application of the above arguments shows that no connection can be in T – a contradiction. □

THEOREM 4.4. *For any network topology, if all routers use* WEIGHTED FAIR QUEUEING *with coordinated weights, then* PROBING INCREASE EDUCATED DECREASE *is incentive-compatible.*

PROOF. By contradiction, assume that there is a coalition T (of size at least one) of manipulative connections that can deviate from PROBING INCREASE EDUCATED DECREASE and better the throughput of each member in the coalition. We shall now prove that no connection can be in T, thus reaching a contradiction.

Consider a connection i. Let α be the minimal value of $\frac{w_i \times c_e}{\Sigma_{r \in K(e)}}$ taken over all edges on i's path. Observe that if i's maximal transmission rate $\overline{f_i}$ is at most α, and all connections are executing PROBING INCREASE EDUCATED DECREASE, then i's throughput is guaranteed to be $\overline{f_i}$. This is because i's (weighted) fair share on each edge on its path is at least α. Because i can never obtain a thoughput that is higher than its maximal transmission rate, we conclude that $i \notin T$. Now, consider the outcome reached after T's deviation from PROBING INCREASE EDUCATED DECREASE. Because $i \notin T$ it must be running PROBING INCREASE EDUCATED DECREASE. Observe that, the same arguments as before still imply that i will eventually get a throughput of exactly $\overline{f_i}$. By PROBING INCREASE EDUCATED DECREASE, we know that i will then increase/decrease its transmission rate so that it exactly matches its thoughput. From that moment onwards, the network is effectively equivalent to a network in which i is removed, and the edges' capacities are updated accordingly. So, from now on we can assume, without loss of generality, that, when all connections are running PROBING INCREASE EDUCATED DECREASE, every connection gets a thoughput that is strictly bigger than its maximal transmission rate.

Let e be the link in G for which the expression $\frac{c_e}{\Sigma_{r\in K(e)} w_r}$ is minimized. Let β denote this value. For every $i \in K(e)$, WEIGHTED FAIR QUEUEING guarantees that, regardless of what the other connections do, i ican obtain a fair share of at least $w_i \times \beta$ on each edge along its path. This implies that every $i \in K(e)$ that executes PROBING INCREASE EDUCATED DECREASE is guaranteed to get a throughput of at least $w_i \times \beta$. In addition, every $i \in K(e)$, that is also in T, is guaranteed (by the definition of T) to get a thoughput that is strictly bigger than $w_i \times \beta$. However, observe that $\Sigma_i(w_i \times \beta) = c_e$. This implies that no $i \in K(e)$ can be in T (for otherwise the capacity of e would be exceeded). Now, if all connections in $K(e)$ are not in T if follows that they are executing PROBING INCREASE EDUCATED DECREASE. Therefore, each $i \in K(e)$ will eventually achieve a throughput of $w_i{\cdot}\beta$, and increase/decrease its transmission rate until it exactly equals its throughput. We conclude that after some time goes by, the network is effectively equivalent to a network without connection i, and in which the capacities of the edges on i's path are updates accordingly.

A repeated application of the above arguments shows that no connection can be in T – a contradiction. $\square$

5. RELATED WORK

Incentive-compatibility has been extensively studied in the context of interdomain routing [4, 6, 12, 18]. In contrast, to the best of our knowledge, this issue has received little, if any, attention in the context of congestion control. Other game theoretic aspects of congestion control have been studied: The *price of anarchy* [17] induced by selfish end-host behavior was examined in [1], where the degraded performance of the network in Nash equilibria, compared with the offline optimal solution, was quantified. There has also been much work on fairness in congestion control [2, 23]. Unlike these works, we are not concerned with the "quality" of the outcome reached by congestion control dynamics, but in guaranteeing that compliant behavior in the convergence process itself be in the best interest of each connection. Our work thus falls within the framework of *distributed algorithmic mechanism design* [5, 4, 7, 21].

A long line of research has studied the dynamic properties of congestion control protocols, including [3, 8, 9, 10, 11, 13, 15, 14, 16, 20, 19, 24]. The majority of these study the case of a single congested router with multiple flows, or of multiple congested gateways, with no cycles and hence no feedback effects [8, 3]. The topology in [16] has a cycle and hence can demonstrate feedback. The authors of [16] solved numerically for a fixed point but did not analyze theoretically the existence of such a point, or convergence to it.

We are unaware of any work analyzing the case of convergence due solely to interactions between the routers or switches themselves, with fixed-rate senders.

6. CONCLUSION

This paper developed a partial characterization of when congestion control schemes can guarantee convergence, incentive compatibility, and efficiency, leaving several directions for future work.

An apparently nontrivial problem which we leave open is to determine whether DROP TAIL converges in the general case. It would also be possible to analyze other queueing policies, such as RANDOM EARLY DETECTION (RED) and FAIR RANDOM EARLY DROP (FRED) [22].

While we have given sufficient conditions and necessary conditions for incentive compatibility and efficiency, they are not tight. A very interesting direction would be to derive conditions that are both sufficient and necessary, or at least to narrow the gap between the two sides. It would also be interesting to see whether in doing so, qualitative differences in convergence remain related to incentive compatibility, as we have demonstrated in the difference between local queueing policies (which can converge only asymptotically and cannot guarantee both incentive compatibility and efficiency) and WFQ and SPQ (which converge exactly and can guarantee both properties).

Finally, our fluid flow model is a significant simplification of real networks. This could be addressed by generalizing our model to permit mixed timescales (rather than assuming that end-hosts react more slowly than routers), and with an experimental evaluation.

7. REFERENCES

[1] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou. Selfish behavior and stability of the internet: a game-theoretic analysis of tcp. *SIGCOMM Comput. Commun. Rev.*, 32(4):117–130, 2002.

[2] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*, pages 1–12, New York, NY, USA, 1989. ACM.

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking Research and Experience*, pages 3–26, October 1990.

[4] J. Feigenbaum, C. H. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1):61–72, 2005.

[5] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer System Sciences*, 63(1):21–41, 2001.

[6] J. Feigenbaum, V. Ramachandran, and M. Schapira. Incentive-compatible interdomain routing. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 130–139, New York, NY, USA, 2006. ACM.

[7] J. Feigenbaum, M. Schapira, and S. Shenker. *Distributed Algorithmic Mechanism Design. A chapter in 'Algorithmic Game Theory"*. Cambridge University Press, 2007.

[8] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. *Computer Communication Review (CCR)*, 21(5):30–47, October 1991.

[9] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 2: two-way traffic, 1991. Unpublished manuscript.
[10] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE ACM Transactions on Networking*, 7(4):458–472, 1999.
[11] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.
[12] S. Goldberg, S. Halevi, A. D. Jaggard, V. Ramachandran, and R. N. Wright. Rationality and traffic attraction: incentives for honest path announcements in BGP. In *SIGCOMM*, pages 267–278, 2008.
[13] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.
[14] F. Kelly. Mathematical modelling of the internet. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited – 2001 and Beyond*, pages 685–702. Springer-Verlag, 2001.
[15] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommuncations*, 8:33–37, 1997.
[16] T. Kelly, S. Floyd, and S. Shenker. Patterns of congestion collapse, 2003. Unpublished manuscript.
[17] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.
[18] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *STOC 08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 57–66, Victoria, British Columbia, Canada, May 2008.
[19] D. Lin and R. Morris. Dynamics of random early detection. *SIGCOMM Comput. Commun. Rev.*, 27(4):127–137, 1997.
[20] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle. Dynamics of tcp/red and a scalable control. In *In Proceedings of IEEE INFOCOM 2002*, pages 239–248, 2002.
[21] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1):166–196, 2001.
[22] R. Oliveira, B. Zhang, D. Pei, R. Izhak-Ratzin, and L. Zhang. Quantifying path exploration in the Internet. In *Proc. Internet Measurement Conference*, October 2006.
[23] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1(3):344–357, 1993.
[24] B. Raghavan and A. Snoeren. Decongestion control. In *Proceedings of the Fifth Workshop on Hot Topics in Networks (HotNets-V)*, pages 61–66. ACM Press, November 2006.
[25] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *Proc. SIGCOMM*, 1998.
[26] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, 32(1):1–16, March 2000.

APPENDIX

A. PROOF OF THEOREM 3.4

PROOF. First, notice that if there is no directed cycle in the dependencies graph, then G_d induces a partial order over the nodes, and every flow on every edge stabilizes immediately after all of its predecessors in this order have stabilized. Naturally, the first nodes in this partial order are stable by definition since they are determined by the (constant) influx of each connection into the network.

Now, if there is a unique cycle in G_d, then any node that is not part of the cycle belongs to one of two groups: nodes that can be reached from the cycle and thus come "after" the cycle, and nodes that come "before" the cycle, and cannot be reached from the cycle.

The nodes that are before the cycle form a directed acyclic graph and all their dependencies are also nodes that come before the cycle. Therefore, from some point on, these values f^i_e before the cycle will converge.

For the rest of the proof, we shall consider any vertex in G_d that is not in the cycle but has a directed edge into the cycle as if it is constant (i.e., its value has converged).

The nodes after the cycle do not influence the convergence of the cycle itself (since the cycle is unique and there are no more feedback loops), and so we will not consider them in the next part of the proof that concerns the convergence of the feedback cycle itself. Once the cycle converges, nodes after the cycle converge as well. They form a directed acyclic graph that has dependencies only in the cycle itself, and in the set of nodes before the cycle (both of which have converged at this point).

Let $\vec{f} = (f^i_e)_{i,e}$ denote some flow configuration for the entire network, and let $\vec{f^*} = (f^{i*}_e)_{i,e}$ be another such configuration. We define the distance between these two configurations to be:

$$d(\vec{f}, \vec{f^*}) = \max_{i,e} |f^i_e - f^{i*}_e|$$

Our goal will be to show that for any fair activation sequence of the edges, from some point on, the distance between $\vec{f}$ and $\vec{f^*}$ approaches 0. This will immediately imply that any flow $\vec{f}$ approaches the constant fixed point flow $\vec{f^*}$ (the proof is correct for any pair of flows). We shall treat all nodes leading into the cycle as constant, while all nodes reachable from the cycle (that do not belong to it) will be ignored for now.

let us observe two consecutive vertices in the cycle: $f^i_{e1} \to f^j_{e2}$. The next lemma shows that an update of the second vertex will usually have a smaller distance than the pervious edge.

LEMMA A.1. *Let $\vec{f}, \vec{f^*}$ be two flow states in the network. If the network topology contains only a single cycle, then for any two consecutive nodes on the cycle in G_d, f^i_{e1}, f^j_{e2}, after node f^j_{e2} is updated:*

- *If the nodes are both from the same flow, and the edge e_2 is un-congested: $|f^j_{e2} - f^{j*}_{e2}| = |f^i_{e1} - f^{i*}_{e1}|$*
- *Otherwise, $|f^j_{e2} - f^{j*}_{e2}| < \gamma \cdot |f^i_{e1} - f^{i*}_{e1}|$ for some $0 < \gamma < 1$*

PROOF OF LEMMA. To abbreviate, we shall denote the two vertices f^i_{e1}, f^j_{e2} by f_1, f_2 correspondingly. There are two cases:

CASE I: $i \neq j$ and the vertices belong to different flows. We denote by f_p the predecessor node of f_2 that does belong to the same flow, and by k the sum of all other flows that affect f_2 directly. Note that both f_p and k are constants and do not change between $\vec{f}$ and $\vec{f^*}$

- if both f and f^* are congested:

$$|f_2 - f_2^*| = \left| \frac{f_p}{f_1 + f_p + k} \cdot c - \frac{f_p}{f_1^* + f_p + k} \cdot c \right| =$$

$$= \frac{c}{(f_1 + f_p + k) \cdot (f_1^* + f_p + k)} \cdot |f_p \cdot f_1^* - f_p \cdot f_1| =$$

$$= \frac{c}{f_1 + f_p + k} \cdot \frac{f_p}{f_1^* + f_p + k} \cdot |f_1^* - f_1| <$$

$$< |f_1^* - f_1|$$

- if only one of the two flows is congested (w.l.o.g. this flow is $\vec{f}$) then $f_2^* = f_p$ and:

$$|f_2 - f_2^*| = \left| \frac{f_p}{f_1 + f_p + k} \cdot c - f_p \right| =$$

$$= \left| \frac{f_p \cdot c - f_p^2 - f_p \cdot f_1 - f_p \cdot k}{f_1 + f_p + k} \right| =$$

$$= \frac{f_p}{f_1 + f_p + k} \cdot |c - f_p - f_1 - k| <$$

$$< |c - f_p - f_1 - k|$$

However, because one flow is congested and the other is not we have:

$$f_1^* \leq c - f_p - k < f_1$$

which now implies:

$$|f_2 - f_2^*| < |f_1 - c + f_p + k| = f_1 - c + f_p + k \leq$$

$$\leq f_1 - f_1^* = |f_1 - f_1^*|$$

- if both flows are un-congested then $f_2 = f_2^* = f_p$ and so

$$|f_2 - f_2^*| = 0 \leq |f_1 - f_1^*|$$

CASE II: $i = j$. I.e., the vertices belong to the same connection. Again, we denote f_{e1}^i, f_{e2}^j by f_1, f_2 to abbreviate, and also let us denote by k, the sum of all other flows (besides f_1) that directly affect f_2. Once again, let us observe the following subcases:

- both flows are congested.

$$|f_2 - f_2^*| = \left| \frac{f_1}{f_1 + k} \cdot c - \frac{f_1^*}{f_1^* + k} \cdot c \right| =$$

$$= \frac{c}{(f_1 + k)(f_1^* + k)} \cdot |f_1 \cdot k - f_1^* \cdot k| =$$

$$\frac{c}{f_1 + k} \cdot \frac{k}{f_1^* + k} \cdot |f_1 - f_1^*| < |f_1 - f_1^*|$$

- only one flow is congested (w.l.o.g this flow is $\vec{f}$).

$$|f_2 - f_2^*| = \left| \frac{f_1}{f_1 + k} \cdot c - f_1^* \right|$$

Now notice that because one flow is congested and the other is not we have:

$$f_1 + k > c \quad ; \quad f_1^* + k \leq c$$

from this we can derive:

$$f_1 > f_1^* \quad \text{and} \quad c - f_1^* > k \quad \text{which gives:}$$

$$(c - f_1^*) \cdot f_1 > k \cdot f_1^* \;\rightarrow\; f_1 \cdot c > f_1^* \cdot f_1 + k \cdot f_1^* \;\rightarrow$$

$$\frac{f_1}{f_1 + k} \cdot c > f_1^*$$

And so we have:

$$|f_2 - f_2^*| = \left| \frac{f_1}{f_1 + k} \cdot c - f_1^* \right| = \frac{f_1}{f_1 + k} \cdot c - f_1^* <$$

$$< f_1 - f_1^* = |f_1 - f_1^*|$$

- both flows are un-congested. In this case, $f_2 = f_1$ and $f_2^* = f_1^*$, and so we have:

$$|f_2 - f_2^*| = |f_1 - f_1^*|$$

Notice that this is the only case in which a strict inequality was not achieved.

At every one of the in-equalities we are in fact able to get an upper bound on the error of the form $|f_2 - f_2^*| < \gamma \cdot |f_1 - f_1^*|$ for some $< 0\gamma < 1$. This requires bounding terms a bit more carefully than what was done in the proof above. The main idea is that from some point in time and onwards, all flows on all edges are strictly greater than zero, and can be bounded from below. This implies that terms such as $\frac{k}{f_1^* + k}$ can be bounded from above by γ. □

Given Lemma A.1, we can see that the maximal distance between the two flows over the cycle in G_d never increases. In fact, because each flow on its own does not contain loops, it is impossible that all edges in the cycle are between vertices from the same connection, and for some edge in the cycle the difference between the flows must strictly decrease. After this edge is activated, the reduced distance propagates along the cycle, until it loops back. During the next activation of that edge, the distance will be decreased even further.