The Role of End-to-End Congestion Control in Networks with Fairness-Enforcing Routers

John McCullough, Barath Raghavan, and Alex C. Snoeren
University of California, San Diego
{jmccullo,barath,snoeren}@cs.ucsd.edu

ABSTRACT

Traditionally, capacity on the Internet has been allocated between competing flows through a distributed fairness calculation implemented by end-to-end congestion control protocols like TCP. Increasingly, however, network operators are deploying fair queuing and other forms of router-based enforcement mechanisms to prevent greedy or misbehaving end points from consuming more than their fair share of the network's capacity. In environments where fairness is enforced by the network itself, it seems worthwhile to reconsider the role of the congestion control protocol. In particular, we ask if it might be both safe and sensible in the long term for self-interested senders to send at rates that exceed the capacity of the network. Through simulation, we identify and quantify the source of inefficiency in this regime, which we term *zombie* packets. Surprisingly, we show that such aggressive mechanisms are not only tenable in a wide variety of network structures, but, combined with effective use of erasure coding, they can avoid creating zombie packets and achieve throughputs that approach optimal.

1. INTRODUCTION

The Internet is fundamentally a shared medium, where individual end hosts compete to receive their 'fair' share of the capacity between themselves and their chosen remote end points. Historically, the rate allocation problem has been solved in a distributed fashion, through end-to-end congestion control protocols like TCP. In particular, end hosts cooperate with each other to empirically determine and maintain an appropriate sending rate in an effort to promote both global efficiency and local fairness. Indeed, the IETF (Internet Engineering Task Force) mandates that all proposed new transport protocols employ a "TCP-friendly" end-to-end congestion control mechanism [13].

Increasingly, however, researchers and operators have observed that individual end hosts deviate from the prescribed altruistic behavior, and instead attempt to greedily increase their share of the network capacity [1, 37], resulting in unfair rate allocations. If an application can tolerate losses induced by over-sending, there is little to be gained by conformance. Fountain codes [25, 30], for example, famously highlighted the feasibility of loss-tolerant transport for broadcast and bulk data transmission [8, 9]. Hence, many of today's networks have deployed router-based fairness enforcement like fair queuing [10, 39] to prevent such traffic from overburdening the network.

We observe that in such networks there are actually two separate control loops computing an individual flow's fair share: one at the routers, and another at the end points. Clearly, the routers' enforcement takes precedence—they can simply drop packets they deem excessive—so the traditional end-host based fairness computation, inherent in congestion control protocols like TCP, is redundant. Hence, in this paper we reconsider the role of end-to-end congestion control in networks with fairness-enforcing routers. In particular, we ask whether end hosts can safely ignore network-wide fairness concerns and focus entirely on their own selfish concerns like throughput.

TCP-friendly protocols are well known to make inefficient use of network capacity, due in part to their congestion-avoidance behavior [18]. We consider transport protocols that exceed the capacity of the network in steady state. Such an approach (sometimes referred to as a *fire-hose* [39]) is often dismissed in the literature due to the potential for congestion collapse—a condition in which the network is saturated with packets but total end-to-end goodput is low. Yet Williamson and Cheriton showed that over-sending can in fact be the optimal strategy in networks with a particular class of fairness-enforcing routers that provide feedback to the senders about demand [40]. Moreover, recent studies with toy topologies have shown that over-sending can even be efficient without feedback in certain situations [6].

In contrast to previous studies, our concern is not with toy networks [6, 14], nor with protocols that require senders to respect router feedback [19, 40]. Instead, we are interested in whether *selfish sending* causes significant, *fundamental inefficiency* due to loss in realistic topologies when *fairness is enforced by routers*. Congestion collapse occurs only under two conditions: if receivers are unable to deal with high loss (so-called *classical* congestion collapse), or if the network topology is such that packet drops occur deep in the network, thereby consuming network resources that could be fruitfully consumed by other flows [14]. The first concern can be addressed by applying efficient erasure coding.

We seek to understand the properties of network structure and traffic demand that give rise to the second condition—in particular, where and when drops occur in networks with router-based fairness enforcement.

A definitive answer is likely to be elusive and certainly requires a more comprehensive and in-depth study than can be reported here. Hence, we restrict ourselves to the more modest goal of considering a small set of intra-domain topologies with the hope of identifying aspects of network topology and traffic demands that impact the efficiency of fire-hose protocols. Following common practice in congestion control literature [1, 14, 22, 41], we begin by considering steady state behavior, noting that studies on toy topologies show that over-sending can be efficient on toy toplogies even when considering flow dynamics [6]. We defer studies of inter-AS topologies and dynamic traffic demands to future work.

As we show, naïve fire-hose-style protocols do indeed make inefficient use of the network. Fortunately, we also find that simple, selfish sender adaptations can dramatically reduce this inefficiency. Our mechanisms perform favorably to Williamson and Cheriton's protocol, without the need for router feedback nor for senders to explicitly compute a flow's fair share. Instead, senders make simple adaptations to enhance the efficiency of their own flows. In contrast to fairness—which it is not at all clear that end hosts have an inherent incentive to achieve—throughput maximization is a rational goal for a self-interested sender to seek.

Our study makes the following key contributions:

- We identify and quantify the cause of (non-classical) congestion collapse using simulation. Previous work identified "dead" packets as the main cause [14]; we refine this definition to a more specific culprit that we term "zombie" packets.

- We design a set of self-interested, aggressive sender policies that significantly decrease the rate of zombie packets in all network topologies we study, including both synthetic and actual ISP backbone graphs.

- We study the robustness of our results to traffic demand skew, and show that our aggressive sender policies rapidly approach the optimal steady-state, fair-share network allocation as demand variation increases.

The remainder of this paper is organized as follows. We begin with a brief review of related work in Section 2. Section 3 explicitly states our assumptions and models of sender and router behavior. We present our simulation methodology in Section 4, and introduce the distinction between dead and zombie packets. Section 5 develops several simple sender models that reduce the impact of zombies on flow throughput. We evaluate the efficiency of these mechanisms using various synthetic network topologies (Section 6) and traffic demands (Section 7), and two real ISP backbones (Section 8). Finally, we consider whether our algorithms are outperformed by those that consider router feedback by comparing with Williamson and Cheriton's protocol in Section 9.

2. RELATED WORK

The ARPANET experienced a series of congestion collapses in 1986 and 1987 that led to the pioneering work on congestion control. Nagle initially observed that networks with large buffers were especially susceptible [32]; later, Floyd and Fall concluded that this phenomena—which they term classical congestion collapse—occurs when the network is busy forwarding packets that are duplicates or otherwise irrelevant upon arrival at their destinations [14]. Jacobson alleviated the problem in the ARPANET by implementing end-to-end congestion control in TCP [15], which seeks to ensure that a network of TCP senders avoids classical congestion collapse by judicious use of ARQ (automatic repeat request). Later enhancements such as SACK [28] ensure that only useful packets will be retransmitted.

Even in today's Internet, the precise set of locations where loss is likely to occur when demand exceeds network capacity is the subject of considerable debate and disagreement. Historically, public peering points such as MAE-EAST and MAE-WEST were among the primary sites of congestion during the 1990s [20]. More recently, however, many researchers have observed that congestion points appear to be dispersed within the network. For example, Akella *et al.* find that bottlenecks are distributed evenly between intra-ISP links and inter-ISP peering links [2]. In contrast, some have made the argument that the network core has or will have practically infinite capacity, and that congestion inside the network is a minor concern, if at all [27, 34].

Despite the success of TCP and its end-to-end congestion control mechanisms, the research community has periodically advocated the delegation of various degrees of computation to the routers. Williamson and Cheriton proposed a model in which routers enforced fairness by varying per-flow drop rates based upon offered loads [40]. Routers provided explicit feedback on total offered load to all clients, thereby enabling end hosts to individually predict their loss rates and select appropriate sending rates. The computational complexity of enforcing per-flow drop rates and providing feedback was significant for the routers of the time, however. As the processing power of routers has increased in recent years, researchers have resumed advocating for transport protocols that leverage router feedback to help determine fair allocations [11, 19]. While many have been shown to out-perform TCP, they fundamentally assume that end hosts abide by the feedback provided by routers and adapt appropriately.

In contrast, we question whether it is necessary for well-behaved senders to adapt their sending rates. In previous work [36], we observed that there are many potential benefits to network architectures based upon non-TCP-friendly congestion control—including increased stability, enhanced robustness to misbehaving senders, and perhaps even decreased router complexity—and suggested departing from the long-held notion of TCP-friendly behavior [29]. Others have made similar suggestions: Matt Mathis, one of the authors of the work that originally defined the notion of TCP-friendliness, speculates that "we will come to realize that TCP-friendly was actually a [sic] untenable position, and has held us back from important innovations" [27]. Following this line of thinking, Bonald *et al.* [6] recently explored the effects of unrestrained senders in simple topologies. Their analytic results indicate that while such an approach can be unstable and inefficient in networks networks without fairness enforcement, those with router-enforced fairness are able to achieve full network utilization. We find that this is not always the case for more complex topologies, yet it is still possible to achieve high efficiency if senders adapt in a strictly self-interested fashion.

3. PRELIMINARIES

Our study starts from the premise that in-network fairness enforcement will be pervasively deployed to limit the impact of greedy or misbehaving senders. Secondly, we assume that senders can leverage erasure coding or other techniques to make effective use of achieved throughput regardless of loss.

3.1 Router enforcement

Researchers have proposed countless approaches to in-network fairness enforcement with various goals in mind. It is therefore critically important to precisely define what form of fairness we expect routers to provide. Practically speaking, fairness in today's Internet is inextricably tied to TCP-friendliness, which is a poor approximation of max-min fairness [16] on a per-flow basis. TCP is well known to be a very poor approximation when faced with multiple bottlenecks and varied round-trip times, and is only exacerbated by recent developments like the pervasive deployment of auto-tuning TCP stacks.

Hence, there has been a growing call to reevaluate whether it is effective or even desirable to have end hosts attempt to calculate and enforce fairness across flows [7, 27].

3.1.1 Max-min flow fairness

While there are many reasonable definitions of fairness, we consider traditional max-min flow fairness in our study. An allocation of bandwidth is max-min fair if it is not possible to increase the rate of any flow without decreasing the rate of another flow with higher rate. We define a "flow" to consist of all packets between two hosts, irrespective of ports or other multiplexing, so no communicating parties can increase their share by adding more flows [4]. Moreover, unlike the traditional max-multi-commodity flow problem, we assume that demand is unsplittable—each flow uses precisely one path from source to destination. An important consequence of max-min fairness is that if a max-min fair allocation exists, it is unique [5].

We opt for max-min flow fairness because link-local max-min fairness is currently implemented by many deployed routers using well-known algorithms, including fair queuing algorithms such as deficit round robin (DRR) [38] and fair dropping algorithms such as approximate fair dropping (AFD) [35]. Note that while a globally max-min fair flow assignment is max-min fair at all links, the converse is not necessarily so—hence, we wish to determine the potential efficiency of networks that enforce max-min fairness on a link-by-link basis.

We model a perfect max-min-fair *dropping* enforcer as it allows us to avoid issues relating to queuing effects. In particular, it ensures that there is no queuing in the network, so over-sending *does not increase* end-to-end delay. Implementations of dropping-based enforcers have a number of additional, practical benefits. AFD, in particular, benefits from $O(1)$ per-packet complexity, no per-class or per-flow state, and appears likely to enjoy widespread deployment in the near future (AFD is expected to ship in the next generation of routers from Cisco). Finally, max-min fairness provides the following useful property.

3.1.2 Pervasive brickwall dropping

Max-min fair routers implement what we term a *brickwall* policy, meaning, given a fixed set of competing demands, an overloaded router provides each flow with a well-defined maximum outgoing share of the link regardless of the amount of demand in excess of its share that a flow offers. Concisely, throughput is *not* proportional to offered load: increasing a flow's arrival rate above its fair share will not increase its delivery rate. Most explicitly designed router fairness mechanisms of which we are aware (including AFD) have this property. Neither FIFO or RED queuing are brickwall policies, however. In particular, a flow can increase its share of the bottleneck by offering more load [24].

Max-min fair brickwall enforcement can be viewed as one endpoint in a more general space of fairness-enforcing dropping policies, with Kelly's proportional fairness at the other extreme [21]. In Kelly's scheme, routers drop packets in proportion to flow arrival rates. Mo and Walrand generalize proportional fairness with α-fairness, which they show converges from Kelly proportional fairness to max-min fairness as α goes to infinity [31]. Williamson and Cheriton's loss-load routers [40] drop flow relative to an offered flow's deviation from aggregate demand in accordance to a penalty factor, k. In such an environment, Williamson shows that, assuming end hosts each optimize their own throughput, routers converge to a steady-state dropping rate of $1/(k+1)$. While higher values of k lead to more efficient network operation, the protocol dynamics take much longer to converge and require every client to numerically solve a k-th order polynomial to calculate its optimal sending rate. While we do not present bounds, Section 5 presents several far simpler sending strategies that achieve similar levels of performance in networks with brickwall routers.

Of course, fairness enforcement is only triggered at a particular router if demand exceeds capacity—i.e., the router is adjacent to a bottleneck link in the network. We will show, however, that many practical topologies have few bottlenecks in the core. Hence, networks may not need fairness enforcement at all routers. Indeed, Bonald *et al.* suggest that uncontrolled senders can function in a drop-tail network if the ratio of the access-link capacity to core capacity is very small [6]. One could enforce such a ratio through a limited deployment of fairness-enforcing routers at the edges [39]. We defer a comprehensive study of the implications of alternative fairness models or partial enforcement to future work; for this study, we assume that every router in the network provides brickwall max-min fairness enforcement.

3.2 Sender behavior

When the onus of fairness control rests with the network, a sender is responsible only for choosing a locally efficient transmission rate and coping with packet loss. While our goal is to study the practicality of strictly self-interested methods of addressing the former task, there remains a wide spectrum of approaches that could be employed for the latter. Hence, as much as possible, we ignore the details of the mechanism employed to recover from packet loss. In particular, we assume that senders employ forward error correction (FEC) to ensure that all delivered packets are useful. Luby's development of *rateless* erasure codes—codes for which a practically-infinite number of unique erasure-coded blocks can be efficiently generated—makes it possible to leverage erasure coding to transmit data streams that are almost impervious to loss [25, 30].

Unlike traditional codes, such as Reed-Solomon, encoding in a rateless fashion can be quite fast—many schemes require only XOR operations—but requires the delivery of a few more coded blocks than the length of the original payload. For most codes, the number of additional blocks depends on the size of the erasure coded payload. One particular approach for utilizing FEC in a fire-hose protocol, Achoo [36], leverages Online codes, whose overheads have been shown to be as low as 3% in practice [30]. In addition, Achoo is robust to loss in the feedback channel as acknowledgments are simply advice of data-receipt and goodput. We postpone consideration of the overheads of specific protocol realizations and focus here on unidirectional flows, neglecting coding overhead when discussing flow throughput.

4. THE IMPACT OF LOSS

With a crisp definition of the problem space, we are now prepared to uncover the source of inefficiency in max-min flow-fair networks with fire-hose-style senders that drive the network past capacity. Any excess demand leads to packet loss, which has two distinct classes of impact: direct and indirect. Packet loss within a flow directly reduces the throughput delivered to the intended receiver. Any potential direct impact of packet loss can be mitigated through rateless erasure coding as discussed above. Indirectly, packet loss may cause inefficient use of upstream network resources: flows other than the one experiencing loss may have been able to put the upstream capacity to better use. We use simulation to quantify this inefficiency.

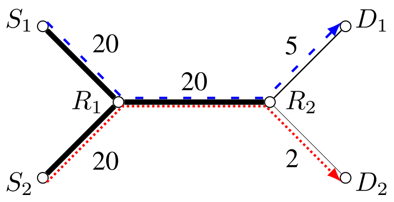


Figure 1: Dropped and dead packets. Each link is labeled with its capacity.

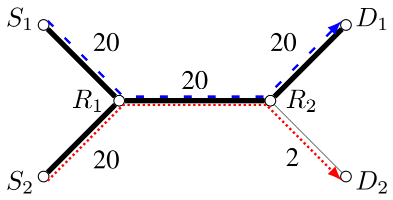


Figure 2: A slightly modified topology leads to zombies.

4.1 Dead and zombie packets

In previous work, Floyd and Fall focus on *dead* packets that are transmitted over several hops, only to be dropped before arriving at their destinations [14]. We call a packet dead if it is dropped at a router other than its source's access router. Figure 1 illustrates this distinction. There are two flows, $S_1 \rightsquigarrow D_1$ and $S_2 \rightsquigarrow D_2$, and each sender is transmitting at a rate of twenty, saturating its access link. Both routers enforce max-min flow fairness, so the throughput of the $S_1 \rightsquigarrow D_1$ flow is five, while the throughput of $S_2 \rightsquigarrow D_2$ is two because they are restricted by the final links. Twenty packets are dropped at R_1, ten from each flow, but none are dead, as R_1 is the access router. At R_2, however, eight more packets are dropped from the $S_2 \rightsquigarrow D_2$ flow. These eight are dead packets, as they traversed the bottleneck link $R_1 \rightarrow R_2$.

4.1.1 Goodput

The issue is not purely the presence of dead packets, rather the presence of dead packets that cause potentially live packets (those that would have otherwise contributed to some flow's goodput) to be dropped. Hence, we introduce a restricted class of dead packets, which we term *zombie* packets, that are deleterious to the network's potential goodput and/or fairness. For example, in Figure 1, if S_2 were sending to D_2 at a rate of only two, eight more packets from the $S_1 \rightsquigarrow D_1$ flow could have traversed the $R_1 \rightarrow R_2$ link but would be subsequently dropped. The goodput of the network would not increase and these dead packets are inconsequential. In contrast, Figure 2 slightly modifies the topology by increasing the capacity of the link from $R_2 \rightarrow D_1$ to 20. Here, flow $S_2 \rightsquigarrow D_2$ continues to receive a goodput of two, but $S_1 \rightsquigarrow D_1$ only increases to 10—not 18, as would be the case in an optimal flow assignment. Hence, in this topology, the eight dead packets in flow $S_2 \rightsquigarrow D_2$ being dropped at R_2 are not only dead, but also zombie packets, as they are preventing $S_1 \rightsquigarrow D_1$ from achieving its optimal rate. Zombie packets are the *only* cause of inefficiency.

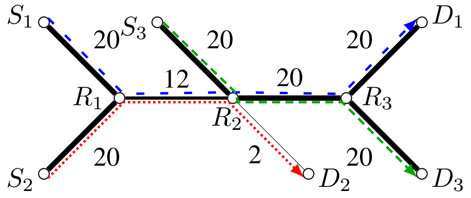


Figure 3: Zombies do not always hurt global goodput.

More formally, we consider a network $G = (V, E)$, with k flows defined by $K_i = (s_i, t_i, d_i)$ where s_i and t_i are the source and sink of flow i, and d_i is the demand (i.e, the amount of flow generated by the source). Let $F_{u,v}$ be the set of flows traversing $(u, v) \in E$, and $f_i(u, v)$ be the amount of flow i traversing edge (u, v). Define MMF(i) to be the demand, d_i of flow i under the ideal, max-min-fair assignment[1], and $L_i(u, v)$ to be the amount of flow i dropped at edge (u, v). The set of flows being under-served at an edge under a flow assignment $\mathcal{F}$ is then

$$\text{Under}(u, v) = \{i \in F_{u,v} | \ f_i(u, v) \leq \text{MMF}(i)\},$$

and the number of dead packets generated at this edge is

$$D(u, v) = \sum_{i \in F_{u,v} \setminus \text{Under}(u,v), s_i \neq u} (f_i(u, v) - \text{MMF}(i)).$$

We calculate the number of zombie packets at an edge as

$$Z(u, v) = \sum_{i \in \text{Under}(u,v)} L_i(u, v).$$

We assign zombie packets in proportion to the number of dead packets caused by a particular flow. That is, for each flow $i \in F_{u,v} \setminus \text{Under}(u, v)$, the number of zombie packets generated at (u, v) is given by

$$Z_i(u, v) = \left(\frac{f_i(u, v) - \text{MMF}(i)}{D(u, v)} \right) \cdot Z(u, v).$$

4.1.2 Fairness

Zombie packets arise when flows compete at a bottleneck (and receive their max-min allocation) while their ultimate throughputs are different from one another. We observe, however, that the occurrence of zombies, while indicating a deviation from the optimal max-min fair allocation, need not lead to a decrease in total network goodput. Figure 3 further extends our running example topology to demonstrate this case. In the optimal max-min fair flow assignment, $S_2 \leadsto D_2$ continues to have a goodput of two, $S_1 \leadsto D_1$ achieves ten, and $S_3 \leadsto D_3$ also obtains a throughput of ten, for a total network goodput of 22 (and Jain's fairness index [17] of 0.79). When senders drive their access links to capacity, however, $S_1 \leadsto D_1$ achieves a goodput of only six in the resulting max-min fair allocation (due to zombies at R_2), but $S_3 \leadsto D_3$ increases to fourteen, leaving the network goodput unchanged (but decreasing Jain's index to 0.68). It is unknown whether there are likely to be other flows that make use of the "wasted" capacity in general [39]. We address this question through simulation.

[1]Under an ideal flow assignment with no loss, $f_i(u, v)$ is the same for each edge flow i traverses.

4.2 Evaluation methodology

To facilitate evaluation at scale, we implemented a flow-based simulator that models perfect max-min link fairness. In contrast to previous efforts that consider random arrivals of finite flows in small networks [6], our simulation models steady state as the behavior of flow dynamics depends critically on the details of an actual protocol which we explicitly avoid modelling. Our simulator takes as input a topology annotated with link capacities and a set of flow demands. Each flow is routed via its shortest path (or actual route in the case of the real topologies used in Section 7) from source to destination, and is subject to no notion of propagation, transmission, or queuing delay (recall that our model assumes dropping—not queuing—routers). At each link, we derive the output flow allocation using a max-min apportionment of the input flows.

Because our simulator does not model packets *per se*, we cannot use packets as the quantity of dead traffic. Instead, we consider the total quantity of dead "flow"—a unit-less quantity where access links are typically assigned a capacity of one flow unit. That is, the maximum possible goodput of a network is equal to the number of access links. If two flows in such a network compete fairly on a link with capacity one, each will drop half of a flow unit. More generally, a given flow has demand minus goodput units of dead flow. For ease of exposition, however, we will continue to refer to this quantity as dead "packets."

The prevalence and location of dead packets in any given network depends on traffic demands. For simplicity, we assume that each source has an infinite amount of data to send. Note that, when using rateless erasure coding, a source with *any* data to send can have infinite data to send. The flow will complete at some point, of course, but flow dynamics are outside the scope of this study. Initially, we assume senders have uniform demand, but relax this assumption shortly.

4.3 Potential inefficiency

We are now equipped to answer the question of whether and where dead and zombie packets may occur with fire-hose sending and link-enforced fairness. As an initial example, we simulate the effect of a simple, uncontrolled fire-hose protocol by setting all of the flow demands to be infinite, and keeping track of the amount of loss at each router. In particular, we assign the initial flow demands, d_i, proportionally as follows. Let $S(u, v)$ be the set of flows that originate at u ($s_i = u$) and traverse edge (u, v), and $c(u, v)$ be the capacity of link (u, v). Then,

$$\forall (u, v) \in E, i \in S(u, v), d_i = c(u, v)/|S(u, v)|. \qquad (1)$$

We simulate the operation of each router to calculate outgoing demand and iterate until flow utilization has been calcu-

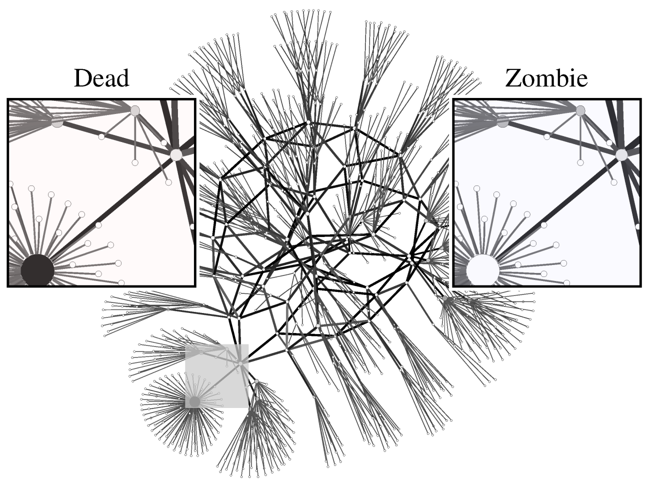


Figure 4: The prevalence of dead and zombie packets caused by fire-hose senders in the HOT with arithmetic capacity assignments.

```
$\forall i \in K,\ d_i = \infty . J = K, R = \emptyset$
while $|J| > 0$
  compute flow through network for demand $J$
  $\hat{i} = \text{argmin}_{i \in J}\ g_i$
  $d_{\hat{i}} = g_{\hat{i}}, R = R \cup \hat{i}, J = J - \hat{i}$
  $\forall (u, v) \in s_{\hat{i}} \rightsquigarrow d_{\hat{i}}, c(u, v) = c(u, v) - f_{\hat{i}}(u, v)$
```

Figure 5: The clamp algorithm computes the optimal max-min far flow assignment.

lated for each edge. The left-hand side of Figure 4 depicts the prevalence of dead packets with fire-hose senders in one run of our simulator on the synthetic HOT topology [23], using an arithmetic progression of link-capacity assignments (see Section 6 for details) and 10,000 flows distributed uniformly at random. With that many flows, almost all links are completely utilized. In the figure, each router's diameter is scaled according to its capacity and shaded according to the amount of dead-packet induced loss occurring at that router; each link is sized according to its capacity and shaded according to its utilization. In both instances, light shades correspond to low values, higher values are darker. We define goodput, g_i, to be the amount of flow i that arrives at the destination, d_i. This simulation run achieved a total network goodput ($\sum_i g_i$) of approximately 125 flow units, while generating almost 650 units of dead packet flow. In this run, almost all the dead packets occur at one particular router with a large attached subtree, called out in the upper left.

While the absolute number of dead packets may seem alarming at first glance, the far more important metric is the degree of inefficiency these dead packets may cause. We use our simulator to compute the globally optimal max-min flow assignment in an iterative fashion through a progressing water-filling algorithm we call *clamp*, shown in Figure 5. Recall K is the set of all flows. Let R be the (initially empty) set of flows for which we have already calculated initial demand assignments, and J be the remainder. We begin by selecting the fire-hose flow with the globally minimum goodput, $\hat{i}$ (ties are broken arbitrarily), and set its initial demand $d_{\hat{i}}$ in the optimal assignment to be the goodput currently achieved in the simulation. We then subtract that demand from the capacity of each link traversed by the flow, remove the flow from the traffic matrix, and rerun the simulator. At each step, we select the flow with minimum throughput and use its achieved value as the flow's demand. After iterating for each flow, we arrive at an optimal set of traffic demands. The flow demands are admissible by construction; Bertsekas and Gallager provide a proof of optimality [5].

In this instance, clamp achieves over 325 flow units of goodput network wide. The decrease in performance of fire-hose does not correspond directly to the number of dead packets, however. The absolute difference in goodput between the fire-hose senders and clamp in this case is approximately 200 flow units, much less than the 650 units of dead packets. In this case, the difference corresponds exactly to the number of zombie packets.

We determine which dead packets in our simulation are actually zombie packets in the following fashion. For a given network topology and traffic pattern, we compare the simulated throughput of each flow with the ideal (the throughput computed by clamp). For each dropped packet in the simulation, we consider whether it causes the flow to decrease below its ideal allocation. Such a drop will occur only if another flow on the same link is over-sending. In other words, another flow will be restricted further along its path, and the packets dropped there are actually zombies. The right-hand side of Figure 4 shows that almost none of the dead packets at the access router impact performance, as the capacity could not have been effectively used by other flows. Regardless of network capacity, adding more flows decreases the relative performance of fire-hose sending; conversely, as the number of flows decreases, so does the number of shared bottlenecks and the performance of fire-hose approaches optimal.

5. REDUCING ZOMBIES

Now that we have precisely identified and quantified the source of inefficiency in our model, we ask whether it is fundamental to all fire-hose style protocols. So far, we have considered the simplest case of oblivious senders who fill their outgoing link with packets equally distributed among all of their destinations. We say that such a sender is exerting equal effort on behalf of each flow. Our simulator models senders with perfect knowledge, in which case there is no benefit to over-sending since the sender is aware of the true bottleneck capacity. In practice, however, over-driving a flow may allow senders to be able to instantaneously take advantage of newly available capacity (due, for example, to the completion of a competing flow or a routing change). While our simulator does not permit us to quantify the potential benefit of this behavior, we can use it to determine

```
$\Delta = \infty$.
while $\Delta > \epsilon$
  compute flow through network
  $\forall i \in K,\ d'_i = d_i$
  update demand assignment $d_i$
  $\Delta = \max_{i \in K} |d_i - d'_i|$
```

Figure 6: Our fixed-point simulation methodology.

any potential negative impact of such behavior. If a sender observes no negative impact from over-sending, it has little incentive not to do so. Conversely, if blindly over-sending hampers the goodput of a sender's own flows, the sender is clearly motivated to adjust its behavior.

We use this insight to devise several slightly more sophisticated send algorithms. The space of potential sending algorithms is vast, and we do not claim to design an optimal one—although their throughput turns out to approach optimal in practice. Figure 6 shows how we use our flow simulator to model the behavior of reactive senders. Initial demand assignments are the same as in the naïve case (Equation 1). After computing the resulting per-flow goodputs, however, we adjust each flow's demand according to the sender algorithm and iterate until a fixed-point is reached. In all cases, we assume senders are aware of the instantaneous goodput of their flows. We detect a fixed-point when no flow's demand is adjusted by more than ϵ in a given iteration.

5.1 Balancing excess capacity

The loss rate—demand minus goodput—of a flow is an approximation of the amount of excess effort a sender is exerting. One intuitive sender behavior is to attempt to spread the excess evenly across all of a sender's flows. A *full-balanced* sender seeks to equalize the loss rate experienced by flows to all of its destinations. In other words, if any flow is seeing proportionally more drops than another, the sender transfers some of its send effort from a flow seeing high loss to a flow seeing low loss. Our simulation of a full-balanced sender implements the demand update step of Figure 6 in the following way:

$$\begin{aligned}\forall (u,v) \in E, i \in S(u,v),\ & \text{set } d_i \text{ such that :} \\ & \forall i,j \in S(u,v), d_i/g_i = d_j/g_j \\ & \text{and } \sum d_i = c(u,v)\end{aligned} \tag{2}$$

This approach behaves identically to the naïve fire-hose sender in the case which there that is a single flow at a sender or $g_i = g_j$ for all flows at the sender.

One potential downside of the full-balanced approach is that large flows will receive a significant amount of excess effort, yet there is no reason to believe that the paths large flows traverse are more likely to experience larger increases in available capacity—which is the main impetus for over-sending, after all—than those over which smaller flows are routed. This observation motivates an alternative, *full-share* sender behavior that distributes any excess capacity evenly (as opposed to proportionally) between flows.

$$\begin{aligned}\forall (u,v) \in E, i \in S(u,v),\ & \text{set } d_i \text{ such that :} \\ & \mathit{slack} = c(u,v) - \sum_{i \in S(u,v)} g_i \\ & d_i = g_i + \mathit{slack}/S(u,v)\end{aligned} \tag{3}$$

In all of our simulations, full-share senders perform almost identically to full-balanced senders, hence we present results only for the latter. Identifying situations in which these two sending policies result in significantly different goodputs is a subject of future work.

5.2 Exercising restraint

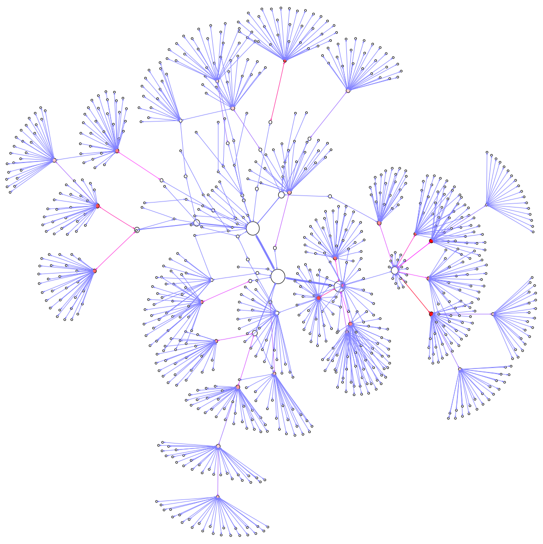
Both of the previous approaches always consume the full capacity of a sender's access link. Blindly filling one's access link, however, may be counter-productive, as it has the potential to inject zombie packets. A *naïve ratio* sender seeks to balance the potential to capture newly available capacity against the harm of injecting zombies by attempting to achieve a certain, fixed loss rate for each of its flows. Said another way, a naïve ratio sender transmits each flow a small, fixed constant, α, faster than the goodput currently reported by the receiver:

$$\forall i \in K,\ d'_i = d_i, d_i = (1+\alpha)g_i. \tag{4}$$

While ratio senders may not saturate their outgoing links if all flows are bottlenecked, they are sure to over-drive their access links if any flow is not. In our simulations, the sender itself implements a brickwall max-min fairness policy on its outgoing link(s). If the total demand on an edge leaving a sender exceeds edge capacity, flow is dropped in a max-min fair manner, which may defeat the sender's attempt to inflate the rate of certain flows relative to others. Hence, we devise a slightly more sophisticated *limited ratio* policy that combines the intents of the full-balanced and ratio strategies to ensure that senders never attempt to over-drive their access links.

$$\begin{aligned}\forall (u,v) \in E, i \in S(u,v),\ & \text{set } d_i \text{ such that :} \\ & \forall i,j \in S(u,v), d_i/g_i = d_j/g_j, \\ & \sum d_i \le c(u,v), \\ & \text{and } d_i \le (1+\alpha)g_i, \\ & \text{where at least one of the above is tight.}\end{aligned} \tag{5}$$

Obviously, the optimal value of α will depend on the dynamics of the network in practice. Clearly, in steady state these strategies will mimic the behavior of clamp when α approaches zero. We have conducted a brief sensitivity study and conclude that steady-state behavior is qualitatively similar in our experiments for positive values of α less than 0.5. We arbitrarily select $\alpha = 0.2$ for the results presented here.


(a) Barabási

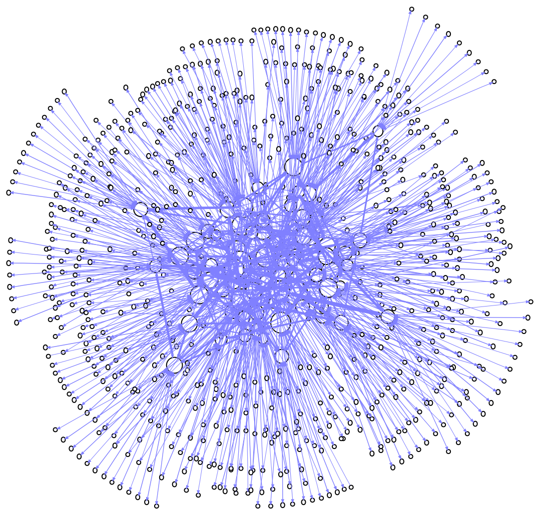

(b) Erdös-Rényi

Figure 7: Sample synthetic topologies with the same number of edge nodes (768) as the HOT topology from Figure 4.

6. NETWORK STRUCTURE

We aim to better understand, for a given network topology, capacity assignment, and demand distribution, whether fire-hose-style sending will induce congestion collapse in networks with max-main fairness enforcement. Our experiments are limited to a single "network" or autonomous system (AS); we do not consider super-topologies of many ASes all connected by direct peering links or connections at public exchanges because we do not have access to such topologies annotated with link capacities and routing tables. In our experiments, we consider both actual and synthetic networks; we start with synthetic networks to study the impact of various aspects of the network independently.

6.1 Topology generation

The composition of the network core directly influences the appearance of dead and zombie packets. We consider three distinct styles of synthetic topologies for the network core: classical Erdös-Rényi random graphs [12]—which we do not expect to represent many real networks—and two more realistic classes: a Barabási-style [3] preferential attachment model and the heuristically optimal topologies (HOT) of Lun Li *et al.* [23], which are thought to be highly representative of a large ISP's router-level topology.

We do not have a generator for HOT graphs. Instead, we obtained a HOT router-level topology generated by Mahadevan *et al.* [26]. This HOT graph consists of 989 links and 939 nodes, 768 of which are edge nodes. In order to facilitate comparison, we seek to create broadly similar network topologies in the other classes. In particular, because the number of edge routers directly limits the potential throughput of the topology, we generate graphs with an equal number of edge routers. Furthermore, because networks are typically structured hierarchically [23], we do not permit edge routers to attach to the core of the hierarchy. Thus, for both Erdös-Rényi and Barabási graphs, we generate instances as follows. First, we generate a random base graph of the desired class. Then, we select the largest connected component and remove any edge routers (nodes of degree one) attached to nodes with more than one non-edge router link. Finally, we add edge routers to nodes at least half of the radius from a central node (determined by maximum betweenness) until we have reached the desired number of edge nodes.

To generate a Barabási-style graph of n nodes and e edges, we iteratively add nodes one by one and probabilistically attach them to (e/n) other nodes with probability in proportion to the node degrees. Inserting a low number of edges results in a hierarchical graph much like the *Preferential Attachment* network generated by Li *et al.* [23]. A high number of edges results in a much denser hierarchy. A low number of edges is shown in Figure 7(a). For the Erdös-Rényi graphs, we insert e edges uniformly at random from among the set of n^2 possible edges. Unlike Li's *General Random Graph* method, we do not expect our Erdös-Rényi approach to obey any power-law distribution. Instead, we expect it to have poor connective behavior for a low number of edges, and to approach a fully connected graph with a high number of edges. We show a graph with a moderate number of edges in Figure 7(b).

6.1.1 Capacity assignment

In order to emulate a real communication network's capacity hierarchy, where the core is typically much faster than the edge links in order to effectively aggregate demand, we assign edge capacities to the synthetic topologies relative to the each link's depth in the graph. A link's depth is computed as the minimum distance from either of its vertices

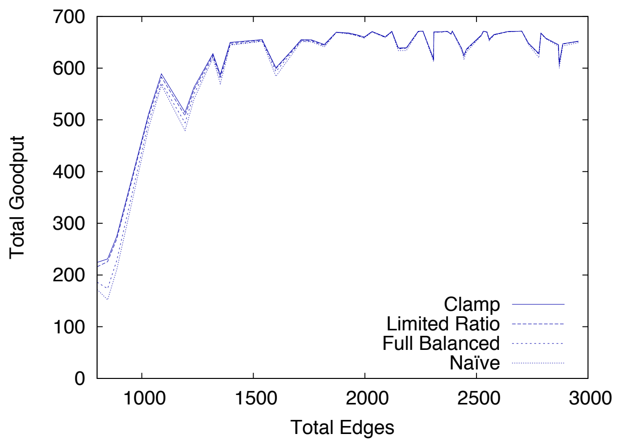


Figure 8: Barabási goodput as a function of core edges; geometric capacity assignment.

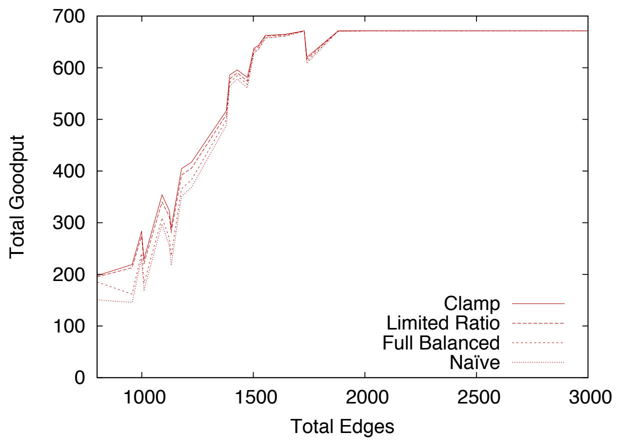


Figure 9: Erdös-Rényi goodput as a function of core edges; geometric capacity assignment.

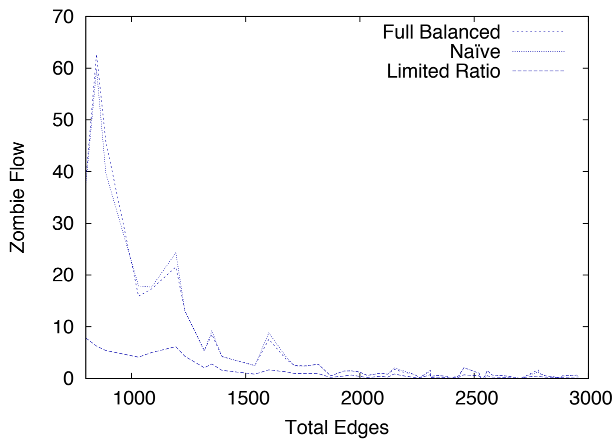


Figure 10: Prevalence of zombie packets in Barabási topologies; geometric capacity assignment.

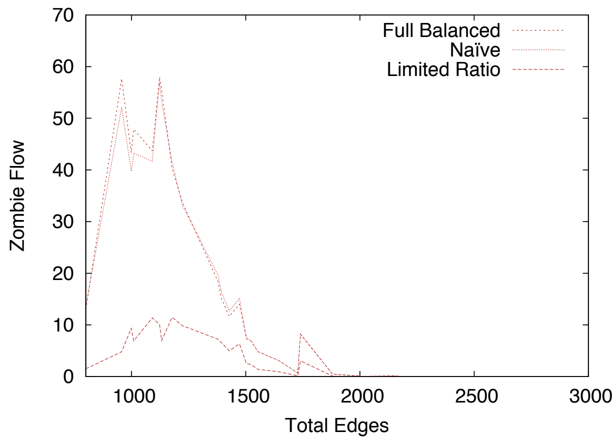


Figure 11: Prevalence of zombie packets in Erdös-Rényi topologies; geometric capacity assignment.

to an edge router. We model two distinct capacity assignment schemes—*arithmetic* and *geometric* progressions—which are likely representative of different types of real networks. For the arithmetic progression, edge capacity is directly proportional to its depth, with leaf edges of capacity one. For the geometric progression, the capacity is directly proportional to $10^{(\texttt{depth}(u,v)-1)}$. Capacity is deliberately unit-less; one can apply arbitrary units.

For the HOT graph, our approach results in a three-tier bandwidth hierarchy. For the geometric progression, a realistic mapping to today's link technologies might be equivalent to 100 Mbps on the edge links, 1 Gbps on the inter-PoP (point-of-presence) links, and 10 Gbps on the core links. An arithmetic progression would then correspond to 100 Mbps at the edge, 200 Mbps at the PoP, and 300 Mbps in the core. In our generated synthetic topologies, we find that similar node counts to the HOT topology create a variance in maximum depth from two to six in respectively dense and sparse graphs. We do not expect these capacity assignment to result in well-provisioned networks or well-engineered networks; rather, they offer a comparison set for evaluation. For instance, with a geometric progression of capacities, it is unlikely that core nodes will be saturated unless they have an exponential number of neighbors. This implies that the bottlenecks will be near the edge. Alternatively, the arithmetic assigned capacities are likely to be under-provisioned with respect to node degree. If the network is extremely well provisioned, the behavior is equivalent to a fully connected core which reduces to a star topology. These capacity assignments serve to produce variations in bottleneck behavior.

6.1.2 Traffic demand

To compare the behavior of the synthetic topologies, we generate uniformly random flows. However, it is important to ensure that the traffic distributions are comparable across topologies. To compare the network-core properties, we want roughly an all-to-all pattern. However, simply choosing edge routers uniformly at random can result in imbalance. In the unlikely case that a one-to-all traffic distribution were created, the throughput would be limited to the single access link. Thus, to achieve comparable traffic demands, we sample sources and destinations uniformly at random without replacement, ensuring that no reflexive flows are generated, and re-initializing the source and desti-

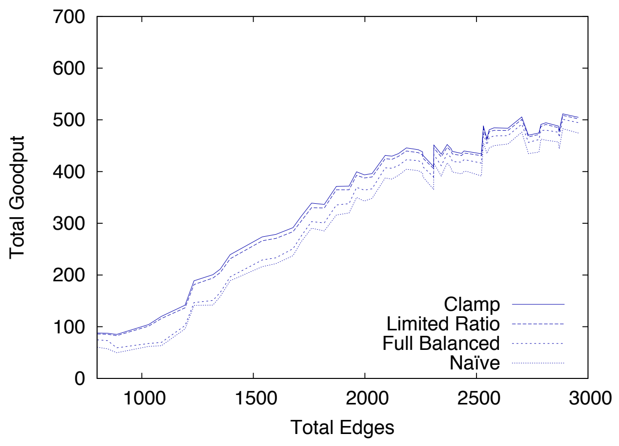


Figure 12: Barabási goodput as a function of core edges; arithmetic capacity assignment.

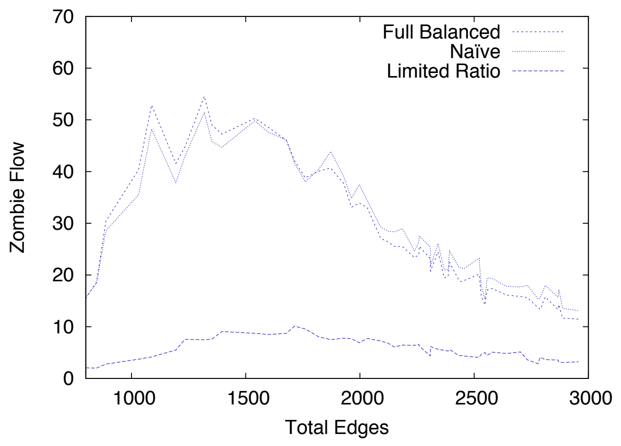


Figure 14: Prevalence of zombie packets in Barabási topologies; arithmetic capacity assignment.

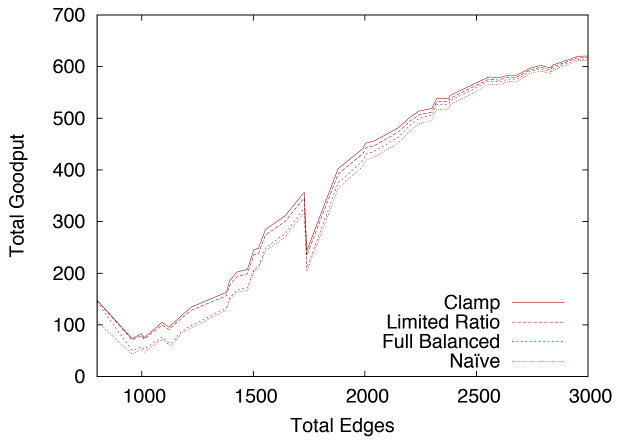


Figure 13: Erdös-Rényi goodput as a function of core edges; arithmetic capacity assignment.

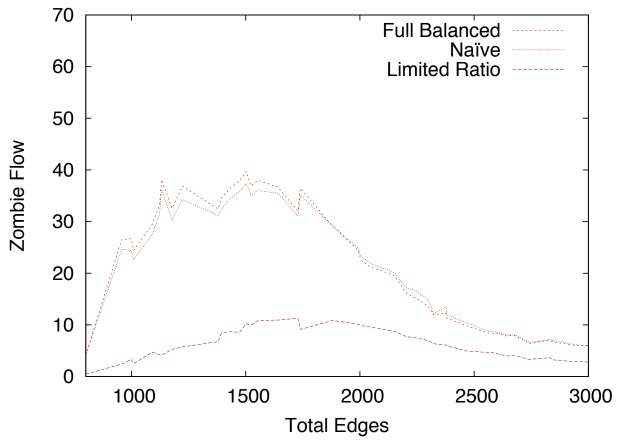


Figure 15: Prevalence of zombie packets in Erdös-Rényi topologies; arithmetic capacity assignment.

nation sets when they are exhausted. This procedure creates a variable all-to-all demand when the number of flows is a multiple of the number of edge routers; if the number of flows is not a multiple of the number of edge routers, there will be slight variation.

Even though differences in the traffic distribution can change the overall flow magnitude, we find that the performance of each algorithm relative to the optimal performance of clamp remains relatively constant. Unless otherwise noted, each data point in the figures that follow represents the mean across ten different flow patterns. Across all of these graphs, the performance of the limited ratio algorithm varies by at most 2% of clamp and both naïve and full balanced remain within 11%. The fractional standard deviations are at most 1% and 6% and 3% for limited ratio, full balanced, and naïve, respectively.

6.2 Impact of core density

The first question we seek to answer is how our various sending algorithms perform as a function of the connectivity of the network core. We construct a series of graphs with varying core density and measure both the goodput and prevalence of zombie packets. Because our HOT graph is a fixed size, we are restricted to Barabási and Erdös-Rényi graphs for this study. For each of these classes, we fix the number of nodes to 768 and vary the requested number of edges. Because we desire similar potential demand, we add edge routers until a specified number is reached. We assign one-and-a-half-times as many flows as there are edge routers.

6.2.1 *Geometric capacity assignments*

We show the results for a geometric progression of capacity assignments in Figures 8 and 9. Somewhat surprisingly, almost all sender strategies achieve the same level of goodput. Within each graph type, however, there are a few trends: First, with small numbers of edges, the core has insufficient bisection bandwidth to satisfy the edge router demands, although the hierarchical structure of the Barabási-style graph provides marginally greater goodput. This trend is expected as a link through a hub is more productive than a link to another edge router. The tendency to connect disparate hubs also contributes to the effect that the full-balanced algorithm achieves greater goodput than limited ratio. This discrep-

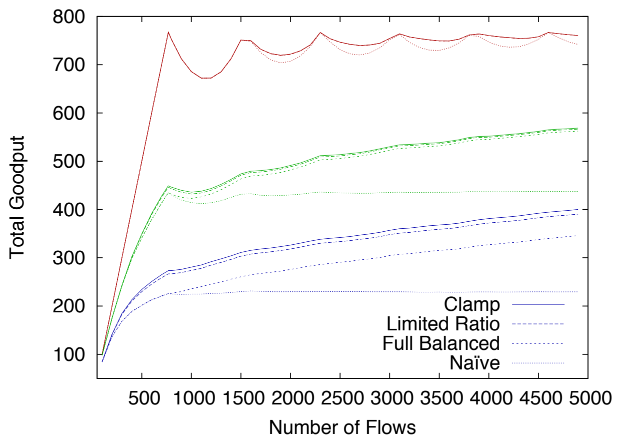


Figure 16: Goodput as a function of the number of flows on Erdös-Rényi, HOT, and Barabási topologies (top-to-bottom, respectively).

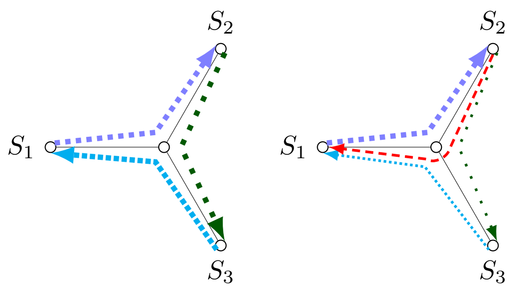
ancy is due to a larger departure from the global max-min fairness as evidenced by the quantity of zombie flow shown in Figures 10 and 11.

6.2.2 *Arithmetic capacity assignments*

One might speculate that the near-optimal performance of the fire-hose sending algorithms is due to the well-provisioned nature of topologies with a geometric progression of capacity assignments. Simulations on corresponding topologies with an arithmetic progression of capacity assignments show this not to be the case. We repeat the previous experiments in an identical fashion, except that each topology is annotated with an arithmetic progression of capacities instead of geometric. Figures 12 and 13 plot the goodput achieved in this case. While the absolute goodput is lower as expected (due to the significantly decreased capacity of the core), all of the sending algorithms perform well, although slightly less than optimal in the case of Barabási topologies. The gap in performance is explained by the increased prevalence of Zombies as shown in Figures 14 and 15 (c.f. Figures 10 and 11). Due to space considerations, we will only present results for a geometric progression of capacity assignments in the remaining sections; results with an arithmetic progression are qualitatively similar.

6.3 Increasing demand

To explore how the number of flows in the graph affects our zombie mitigation techniques, we progressively add flows to fixed instances of all three different styles of graph. We employ a Barabási-style topology with 847 nodes, 857 edges, and 768 edge routers (shown in Figure 7(a)) that mimics the HOT topology. The Erdös-Rényi style graph has the same number of nodes and leaves as the Barabási-style graph, but it includes 318 more edges, and is much better connected than either of the others (Figure 7(b)). In all three cases, we employ a geometric progression of capacity assignments. Figure 16 plots the results for all three types of



(a) Total goodput: 6 (b) Total goodput: 5

Figure 17: A simple star graph with links of capacity two. Starting with three flows (left), adding an additional flow $S_2 \rightsquigarrow S_1$ can decrease the overall goodput (right).

graphs. The top set of lines corresponds to the Erdös-Rényi topologies, HOT is in the middle, and the lower set represents the Barabási results.

The strong connectivity of the Erdös-Rényi graph enables all of the sending strategies to satisfy the full demand of the edge routers. While it appears that the HOT topology is better provisioned than the Barabási-style graph, the performance of the sending algorithms are similar. In particular, in both cases, naïve fire-hose quickly converges to a sub-optimal limit, whereas the other sending techniques track much closer to optimal. Note, however, that full balanced is further from optimal than limited ratio in the Barabási-style graph.

A particularly interesting effect most visible in the Erdös-Rényi style graph is the periodic decrease in overall throughput with an increasing number of flows. This behavior is endemic to any network with locally enforced max-min fairness, and can be explained as follows. Consider a simple star topology with link capacities of two as shown in Figure 17. First, assign three flows: $S_1 \rightsquigarrow S_2$, $S_2 \rightsquigarrow S_3$, and $S_3 \rightsquigarrow S_1$. Each flow achieves the full capacity regardless of sending strategy, resulting in a total goodput of six. If we add a flow $S_2 \rightsquigarrow S_1$, it shares the link to the hub with flow $S_2 \rightsquigarrow S_3$ and they split capacity of the outgoing link. Additionally, $S_2 \rightsquigarrow S_1$ shares the link from the hub to S_1 with $S_3 \rightsquigarrow S_1$. $S_2 \rightsquigarrow S_1$ was already sending at a rate of one flow unit from the earlier link sharing, but $S_3 \rightsquigarrow S_1$ will decrease to one flow unit due to fairness enforcement. This leads to a total goodput of five. Additional flows will lead to an increase in goodput in an oscillatory fashion. As the number of flows increases, the effect's magnitude decreases, leading to the asymptotic behavior as in Figure 16.

7. TRAFFIC SKEW

So far, we have considered uniform demand which is unlikely in most real networks. Actual Internet traffic has been observed to exhibit significant locality, or *skew*. We model skew as a concentration of demand on an increasingly small subset of edge nodes. We replace the uniform-without-replacement flow assignment methodology used previously

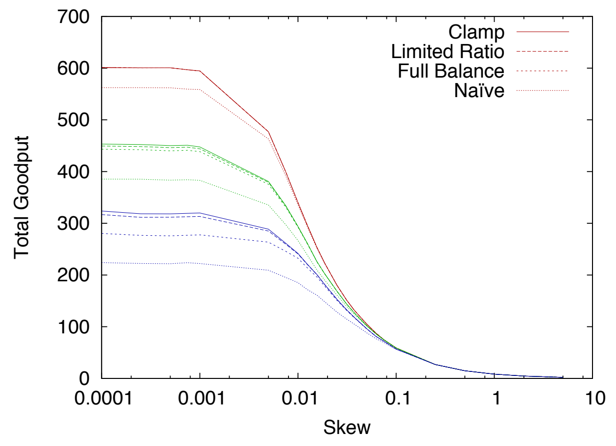


Figure 18: Goodput as a function of traffic skew for Erdös-Rényi, HOT, and Barabási topologies (top-to-bottom, respectively).

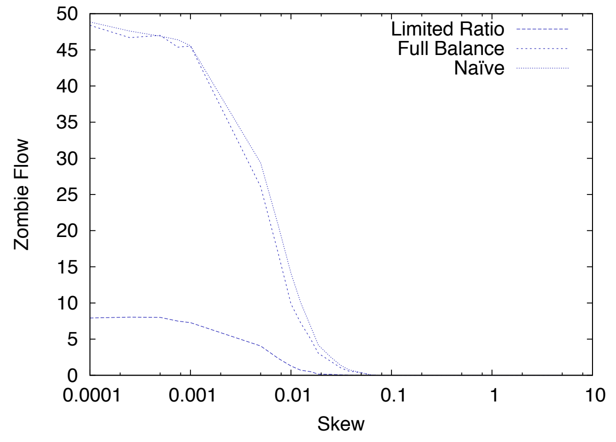


Figure 19: Number of zombie packets as a function of traffic skew for a Barabási topology.

with one in which sources are chosen from an exponential distribution with exponent λ and destinations chosen uniformly at random with replacement. Sampling sources from a Zipfian distribution produces qualitatively similar results; we present only the exponential results for brevity.

Intuitively, a uniform traffic demand will produce the highest goodput, as flows are least likely to share bottlenecks. As traffic matrices become skewed, more and more demand bottlenecks at the access links of a few popular sources, so a smaller portion of the demand will be satisfied. Figure 18 confirms this hypothesis by plotting the achieved goodput of 2,500 flows on the Erdös-Rényi, Barabási, and HOT topologies. Results with a uniform distribution of sources are similar to those with $\lambda = 0.0001$, as can be seen from the 2,500-flow intercept in Figure 16. Note that it is not precisely identical as the sources are chosen without replacement in the earlier experiment.

Figures 19 and 20 show that zombies rapidly dissipate as skew increases for both Barabási and HOT topologies (only the naïve sending strategy generates zombies in the Erdös-Rényi topology, not shown). Indeed, even the naïve strategy

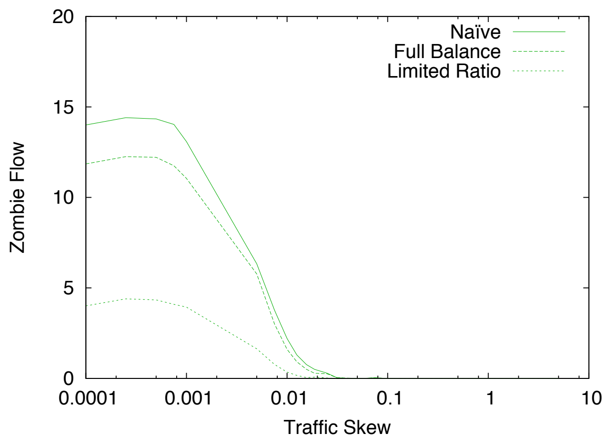


Figure 20: Number of zombie packets as a function of traffic skew for the HOT topology.

ceases to create zombies once the skew surpasses 0.1. While traffic demands obviously vary from network to network, as a point of reference we find that traffic in the GEANT2 network presented in the next section is well modeled by an exponential distribution with exponent 0.5—indicating that zombies are almost inconsequential in our synthetic topologies with arithmetic capacity assignments.

8. REAL NETWORKS

Of course results on synthetic graphs are simply that—synthetic. In order provide some insight into the potential for fire-hose style sending in existing topologies, we consider two separate real—if not necessarily representative—large-ISP PoP (point of presence)-level topologies: an August 2007 snapshot of the GEANT2 pan-European research and education network, and a once-public Level 3 PoP-level map from 2006 annotated with link capacities. The GEANT2 topology consists of a variety of European research networks attached to a core of 21 PoPs connected by a 10-Gbps backbone; we prune the topology to the 35 core links for which we were able to obtain routing and traffic information. The Level 3 topology contains 66 PoPs connected primarily by OC-192 links and secondarily by OC-48 and OC-12.

We obtained actual GEANT2 traffic matrices for August 1, 2007. In those snapshots, the demand does not saturate—let alone over-drive—the network. To generate higher demand that continues to reflect the distribution of the original traffic matrix, we attach leaf nodes to each flow's actual origin and destination and source and sink the simulated traffic at these leaf nodes. (Hence, each leaf node sources and sinks precisely one flow.) The total leaf node access capacity at each original node is identical to the actual capacity of the original node; each leaf node's incoming/outgoing capacity is set in proportion to the fraction its flow represents of the actual node's total original incoming/outgoing demand. We scaled the traffic demands proportionally so that each node's demand is 50 times its capacity. We do not have access to information about the traffic on Level 3's backbone, so we

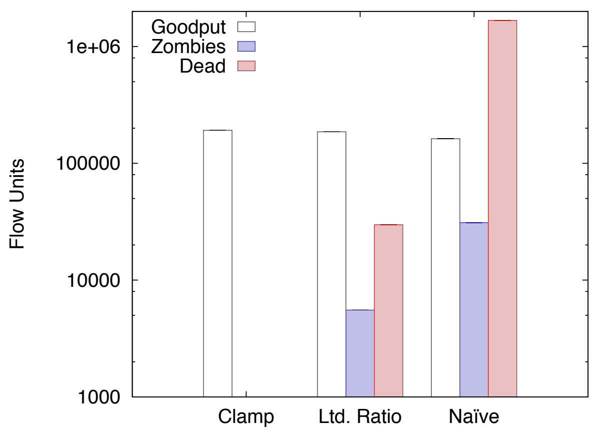


Figure 21: Goodput, zombies, and dead packets for the GEANT topology.

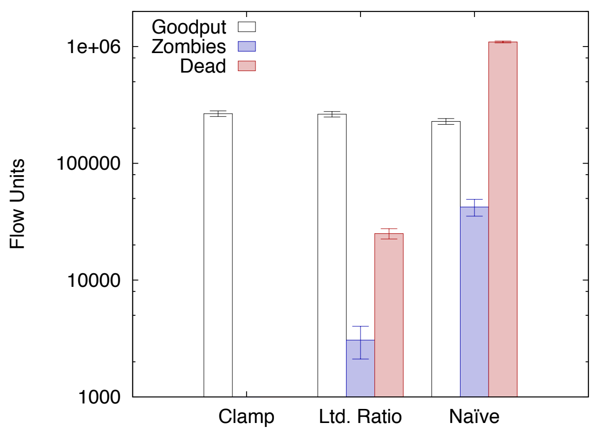


Figure 22: Goodput, zombies, and dead packets for the Level 3 topology.

generated a synthetic traffic matrix following a log-normal distribution as recommended by Nucci *et al.* [33] and augmented the topology in the same manner.

As we show in Figure 21, The optimal max-min fair allocation for the GEANT2 topology achieved by clamp is approximately 192 Gbps. All of the sending strategies are surprisingly close, obtaining goodputs of 186 and 162 Gbps, respectively, for limited ratio and naïve senders. (Recall that all senders source exactly one flow in our construction, so full-balanced senders are indistinguishable from naïve.) The increasing rate of zombies (5.5 and 31 Gbps, respectively) indicates an increasing deviation from the max-min fair allocation, however. The number of dead packets induced by the naïve senders is quite high (approximately 1,675 Gbps), but the vast majority of them are inconsequential.

The results for the Level 3 topology are qualitatively quite similar. Figure 22 shows the mean and standard deviation of ten different traffic distributions. The optimal max-min allocation achieves a mean goodput of approximately 266 Gbps, while limited-ratio and naïve senders achieve an average of 263 and 228 Gbps, respectively. Once again, due to the traffic distribution, full-balanced senders behave as naïve ones, both generating 42.3 Gbps of zombies on 1,095 Gbps of dead packets. Limited-ratio generates slightly fewer of each: 3.1 Gbps of zombies out of 25 Gbps of dead packets.

It is difficult to directly compare the two topologies, as the demand distributions and capacity hierarchies are quite different. For both GEANT2 and Level 3, however, the naïve sender achieved a goodput within approximately 85% of optimal, and our limited-ratio sender strategy is able to close to within at least 3%.

9. FEEDBACK COMPARISON

While we are heartened by the performance of our simple sending algorithms—limited ratio in particular—they are admittedly somewhat *ad hoc*. One might ask if better algorithms exist, or, alternatively, if performance bounds can be obtained. Our sending algorithms belong to the class of algorithms that operate without router feedback; their performance is upper bounded by the performance potential of algorithms that leverage router feedback. We briefly compare against the performance of one such algorithm.

Unfortunately, we are not aware of existing fire-hose algorithms that leverage feedback from max-min brickwall routers. Williamson and Cheriton, however, devised such an algorithm for what they call loss-load routers, which enforce a from of proportional fairness where each flow is dropped in proportion to its deviation from the average [40]. In their approach, routers update senders with aggregate flow statistics every few hundred milliseconds. Based on this information and a network-wide penalty parameter k, senders can compute a loss-load curve for their bottleneck links. A loss-load curve indicates how much loss a sender can expect for any offered rate. To optimize throughput, sender numerically solve a degree-k polynomial until the aggregate statistics reported by the routers settle. Williamson and Cheriton show that, in steady state, flows converge to $(1 + 1/k)$ times their bottleneck rate with an expected loss rate of $1/(k + 1)$.

Intuitively, if loss-load is sending $(1 + 1/k)$-times the path's capacity, then its performance should be very similar to a limited ratio sender with $\alpha = 1/k$. To verify, we implemented the loss-load algorithm in our simulator. We replaced our max-min brickwall routers with loss-load routers and provide feedback in an ideal fashion, i.e., instantaneously. We apply Newton's method to solve the degree-k polynomial; the associated numerical instability necessitated a higher error tolerance to complete the fixed-point computation. Figure 23 shows the fractional mean goodput and standard deviation relative to clamp for ten flow patterns on the Level 3 topology. For each $\alpha = 1/k$ pair, we find that the goodput is nearly identical on individual flow patterns and that the amount of zombie flow is quite similar. However, for other graphs and flow patterns we observe that $\alpha > 1/k$ can achieve better throughput. We find that, as the authors observe [40], loss-load penalizes multi-bottleneck flows. In particular, repeating the skew experiments from Section 7

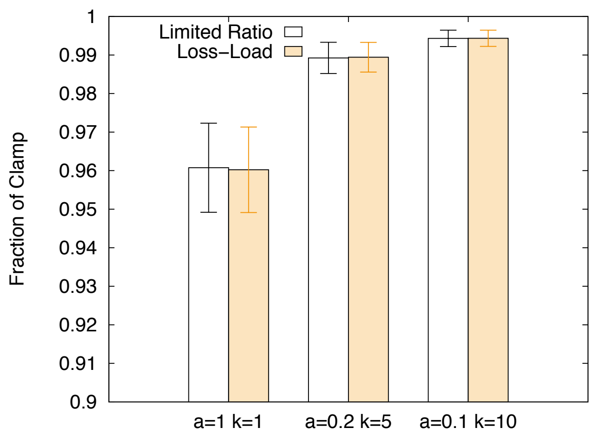


Figure 23: Goodput for Limited Ratio and Loss-Load on the Level 3 topology.

shows that limited ratio with $\alpha = 1$ outperforms loss-load with $k = 10$ for $\lambda > 0.005$. In sum, limited ratio performs comparably to loss-load with a less complex mathematical basis and without explicit router feedback.

10. CONCLUSION

Despite the increasing deployment of fairness-enforcing routers, there has been little work studying the implications for congestion control. Others have studied the effects of oversending on toy topologies and oversending in the presence of router feedback. Our results indicate that for the larger topologies and traffic demands we study, it may be possible for end hosts to completely ignore fairness concerns when selecting send rates, and focus strictly on their own flows' goodput. Our study provides concrete evidence that fire-hose sending, traditionally eschewed as untenable, may be feasible and also identify sending strategies that mitigate inefficiency and approach optimal for realistic network topologies.

11. REFERENCES

[1] A. Akella, R. Karp, C. Papadimitrou, S. Seshan, and S. Shenker. Selfish behavior and stability of the Internet: A game-theoretic analysis of TCP. In *ACM SIGCOMM*, 2002.

[2] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *ACM SIGMETRICS*, 2003.

[3] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Review of Modern Physics*, pages 47–97, 2002.

[4] E. Altman, D. Barman, B. Tuffin, and M. Vojonović. Parallel TCP sockets: Simple model, throughput and validation. In *IEEE INFOCOM*, 2006.

[5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1992.

[6] T. Bonald, M. Feuillet, and A. Proutiere. Is the "law of the jungle" sustainable for the Internet? In *IEEE INFOCOM*, 2009.

[7] B. Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM CCR*, 37(2), 2007.

[8] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *ACM SIGCOMM*, 2002.

[9] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM*, 1998.

[10] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM*, 1989.

[11] D. Ely, N. Spring, D. Wetherall, S. Savage, and T. Anderson. Robust congestion signaling. In *IEEE ICNP*, 2001.

[12] P. Erdös and A. Rényi. On random graphs. I. *Publicationes Mathematicae*, pages 290–297, 1959.

[13] S. Floyd and M. Allman. Specifing new congestion control algorithms. RFC 5033, IETF, Aug. 2007.

[14] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM ToN*, 7(4), 1999.

[15] V. Jacobson. Congestion avoidance and control.

[16] J. M. Jaffe. Bottleneck flow control. *IEEE/ACM Transactions on Communication*, 7(29), July 1980.

[17] R. Jain, D. M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, DEC Research Report TR-301, 1984.

[18] C. Jin, D. X. Wei, and S. H. Low. FAST TCP: motivation, architecture, algorithms, performance. In *IEEE INFOCOM*, 2004.

[19] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM*, 2002.

[20] M. Kaufman. MAE-West congestion, NANOG mailing list. `http://www.irbs.net/internet/nanog/9603/0200.html`, 1996.

[21] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8, 1997.

[22] T. Kelly, S. Floyd, and S. Shenker. Patterns of conjestion collapse, 2003. unpublished.

[23] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet's router-level topology. In *ACM SIGCOMM*, 2004.

[24] D. Lin and R. Morris. Dynamics of random early detection. In *ACM SIGCOMM*, 1997.

[25] M. Luby. LT codes. In *Proceedings of IEEE FOCS*, 2002.

[26] P. Mahadevan, C. Hubble, B. Huffaker, D. Krioukov, and A. Vahdat. Orbis: Rescaling degree correlations to generate annotated internet topologies. In *ACM SIGCOMM*, Kyoto, Japan, 2007.

[27] M. Mathis. Heresy following "TCP: train-wreck". Note to ICCRG (Internet Congestion Control Research Group) mailing list, 2008.

[28] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018, IETF, Oct. 1996.

[29] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM CCR*, 27(3), July 1997.

[30] P. Maymounkov. Online codes. Technical Report TR2002-833, New York University, 2002.

[31] J. Mo and J. Walrand. Far end-to-end window-based congestion control. *IEEE/ACM ToN*, 8(5), 2000.

[32] J. B. Nagle. On packet switches with infinite storage. *IEEE Transactions on Communications*, COM-35(4), Apr. 1987.

[33] A. Nucci, A. Sridharan, and N. Taft. The problem of synthetically generating IP traffic matricies: Initial recommendations. *ACM SIGCOMM CCR*, 35(3), July 2005.

[34] A. Odlyzko. Data networks are lightly utilized, and will stay that way. *Review of Network Economics*, 2(3), 2003.

[35] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Approximate fairness through differential dropping. *ACM SIGCOMM CCR*, 2003.

[36] B. Raghavan and A. C. Snoeren. Decongestion control. In *HotNets-V*, 2006.

[37] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. In *ACM SIGCOMM*, 1994.

[38] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round robin. In *ACM SIGCOMM*, 1995.

[39] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *ACM SIGCOMM*, 1998.

[40] C. L. Williamson and D. R. Cheriton. Loss-load curves: support for rate-based congestion control in high-speed datagram networks. In *ACM SIGCOMM*, 1991.

[41] H. Zhang, D. Towsley, and W. Gong. TCP connection game: A study on the selfish behavior of TCP users. In *IEEE ICNP*, 2005.