Controlling Applications by Managing Network Characteristics

Vahab Pournaghshband, Leonard Kleinrock, Peter Reiher and Alexander Afanasyev
Computer Science Department
University of California, Los Angeles
{vahab,lk,reiher,afanasev}@cs.ucla.edu

***Abstract*—Edge network operators have limited tools to control activities on their networks. This paper examines network dissuasion, a new approach to edge network control, based on controlling the fundamental parameters of the network, such as loss rate, delay, and jitter, with the intention of making particular uses of a network intolerable, while providing acceptable services for approved network uses. We investigate using this technique to prevent use of Voice Over IP (VoIP), while allowing other services. We designed network controls to achieve this goal and performed experiments using both measurements and subjective testing with human beings. We report on the degree of success and discuss the general promise of network dissuasion.**

I. Introduction

Edge network operators often wish to control their networks' activities. In some cases, the owners of the network may want to ensure that non-business activities are not performed on the network. In others, legal issues make avoiding certain activities (such as copyright infringing uses of peer networks) desirable. When resources are limited, owners may want to prioritize network use. It is desirable for a network's owner to have some control over the data flows it carries.

Such a network owner has limited options. Firewall technologies can shut particular ports, restrict access to undesirable IP addresses, or enforce bandwidth limits on customers. If the owner owns the computers attached to the network, he can decree that certain applications are not to be run. If he can be more intrusive, his system administrators can delete or disable offending applications from the network's computers. If he can afford the manpower, his network administrators can observe data flows and provide some control.

These methods have disadvantages. Clever users avoid many firewall restrictions by tunneling prohibited protocols over other protocols or using proxies. By encrypting their traffic, they make the content and even packet headers opaque. If system administrators disable certain programs on their machines, they re-install them after deletion or run them off removable media. If misbehaving users hide their traffic and activities well enough, these control options will not work.

Research on denial of service attacks suggests another approach. Clever attackers deny service to an application by making the network conditions unsuitable for its needs. Brute force denial of service attacks achieve this by using up all the bandwidth, but clever burst attacks can force TCP to misbehave badly without causing much network congestion, and other attacks induce unfavorable network conditions that effectively disable applications.

Can we reverse this observation to benefit network owners by engineering a network's performance to provide good quality of service for applications that the owner wanted to support, but unacceptable quality of service for those he does not want? Naively, we could look for port and protocol numbers, but those are easy to change. But what if the underlying network's characteristics could not reasonably support certain applications due to too much delay or jitter, or too many dropped packets? If too many packets are dropped, then too many packets are dropped, no matter how cleverly they are disguised. We term this technique *network dissuasion*, since it attempts to make undesired use so unpalatable for the typical user that he won't bother to try.

The idea has obvious challenges. Can one, for any arbitrarily chosen "undesirable" application, find network conditions that make its use intolerable, regardless of how users respond? If so, will any other chosen set of desirable applications be relatively undisturbed?

This paper is a preliminary investigation into network dissuasion. We chose a candidate application to treat as undesirable (VoIP) and other applications to support (DNS, web browsing, large file transfer, and interactive keyboard applications). We sought techniques that make VoIP unacceptable, while having modest impacts on the other applications. We report our experiences in designing network conditions of this nature, how we achieved these conditions experimentally, and the measured utility of both types of applications under these conditions. Our results show that, for this set of applications, network dissuasion can work. But it is challenging, and many non-obvious measures must be taken. Much depends on definitions of acceptable service. Thus, while the technique is not impossible, our results suggest it will be challenging to generalize it to a wide range of arbitrarily chosen desirable and undesirable applications.

The paper is organized as follows: Section II presents related work which is followed by background in Section III. Methodology, experiments, discussion, and conclusion remarks are preseneted in Sections IV, V, VI, and VII, respectively.

II. Related work

Much research has addressed controlling traffic. Traffic shaping is particularly closely related, being mainly concerned

with controlling the volume and rate of traffic being sent through a network, often to favor or disfavor particular classes of traffic. This is often accomplished through special treatment of queues at routers, such as leaky bucket algorithms [1], token bucket algorithms [2], CBQ [3], RED [4], and Diffserv [5]. With a different objective, traffic policing [6] focuses on properly classifying data streams to ensure that specified bandwidth limits are applied.

Various techniques are used to conceal network traffic, sometimes to bypass networks or content control. Onion routing [7] and anonymizing web proxies [8] are examples. A variety of tools permit traffic of one type to be tunneled over a different transport or application-level protocol. Tunneling various kinds of traffic over HTTP is common [9], and other protocols, such as DNS, have also been investigated for this purpose [10]. This body of work suggests that clever techniques are often developed and deployed to overcome network surveillance and control. This serves as a motivation for finding new mechanisms to apply controls, particularly mechanisms that do not rely on surface characteristics of packets or flows.

III. Background

An ideal network dissuasion system would permit an arbitrary set of unacceptable applications to be discouraged, while a different arbitrary set of acceptable applications is minimally impacted. To achieve this ideal, one must first demonstrate that applications can be dissuaded at all. Trivially, any application becomes intolerable if the bandwidth is low enough or the delay is long enough, but that may stop all applications. We designed a non-trivial test case to see if we could achieve network dissuasion even in one circumstance. Beyond serving as evidence that the technique could or could not work, we sought insight into the effort required to design network characteristics suitable for dissuading particular applications, and broad insights into whether the technique warranted deeper investigation.

Our test case was to dissuade VoIP while supporting DNS, web browsing, file transfer, and an interactive shell application. This test case was complex enough to be challenging, but the relative fragility of voice traffic (compared to other network activities) suggested that it might be feasible. Also, some operators might wish to discourage VoIP, such as airlines that allow Internet activities while in flight, while still trying to offer premium-cost telephony services.

We first studied how VoIP applications react to bad network conditions. We also had to understand how the desired set of applications responds to such network conditions. These requirements imply that we must have methods to measure the applications' quality of service.

A. VoIP Characteristics

Delay, loss, and jitter are network factors with major negative effects on the quality of voice transmitted over the Internet. In this section we present relevant popular approaches that VoIP applications take to cope with adverse network conditions and discuss how effective these approaches are.

1) Loss: In VoIP terminology, a packet is considered lost if either it never arrives at the receiver or it arrives after its scheduled playback time. Losses are unavoidable on the Internet, and thus VoIP applications must be robust against them. Popular VoIP applications are robust against single losses and indeed some are tolerant of up to three consecutive (or bursty) losses (at 20 ms packet intervals) [11].

2) Application Congestion Control: VoIP applications use proprietary congestion control algorithms to respond to various network conditions. Skype congestion control, for example, primarily uses loss ratio to adjust the sending rate to match the available bandwidth [12]. To adjust the sending rate, the receiver monitors feedback variables such as RTT and loss ratio, and periodically piggybacks this information to the sender in data packets. The sender receives data and adjusts the sending rate and the packet size accordingly [13].

3) Blocked UDP: VoIP applications tend to avoid TCP, due to TCP's retransmission and congestion control mechanisms. Some, however, use TCP when they sense that UDP traffic is blocked. Skype, for instance, builds an overlay network by finding the nearest super node[1] to the blocked node, and then routing all traffic to and from the node behind the firewall through this super node. To minimize the undesirable effects of TCP, only the connection between the super node and the constricted node would use TCP, with the remaining connection using UDP [15].

B. Quality of Service for Network Applications

For our particular experiment to be successful, we should be able to confidently argue that while the permitted applications are running satisfactorily, the voice quality is unsatisfactory. This necessitates a precise definition of the quality of service (QoS) requirements for the following applications:

1) VoIP: There are two widely accepted approaches for measuring voice quality: subjective and objective testing.

Subjective measurements of VoIP QoS use a group of people, called test subjects. Test phrases are recorded, and test subjects listen to them in different environments. In Absolute Category Rating (ACR) [16], test sequences are presented serially and are rated independently on a category scale. Subjects listen to a test sequence and rate the quality of the voice based on their own understanding. The Mean Opinion Score (MOS) quality scale is the most popular ACR test metric. MOS is determined by asking users to listen and rank the quality of voice from 1 to 5 (5 being excellent and 1 indicating bad quality). The arithmetic mean of all responses is calculated; 3.0 and above is considered acceptable quality.

Objective testing measures various observable quantities. Subjective measurements are the benchmark for objective methods, but because they are slow, time-consuming, and expensive, objective testing methods are more popular.

[1] A Skype's super node is any client closest to the Skype client beyond the firewall. Any Skype client with a public IP address, sufficient CPU and memory, and network bandwidth can become a super node [14].

2) Permitted Applications: The chosen interactive applications involve a human requesting a service from a remote server and waiting for the response. For such applications, the primary QoS requirement is a response within a user-acceptable delay. Such user tolerances are subjective and variable. To have some usable basis, we evaluate using acceptable thresholds defined in research performed by others. Service quality is assessed by comparing measured parameter values with the corresponding thresholds. We use *whole-delay*, defined as the delay measured from the conclusion of a user's request until receipt of the entire response.

a) DNS: DNS QoS is particularly vital since poor DNS service could cause total unavailability of other services. For DNS, a whole-delay of at most 4 seconds is required [17].

b) Web: According to Bouch et al. [18], web browsing requires no more than a 60-second whole-delay.

c) FTP: For applications like FTP, long delays are just annoying. Mirkovic et al. [19] suggest that the file transfer whole-delay should not exceed three times the delay experienced without interference.

d) SSH: SSH gives two types of responses: (1) echoing typed characters and (2) generating responses to requests. *Echo-delay* is the delay between a user's typing a character and receiving the echo packet. Nortel Networks [17] suggests 250 ms as a maximum acceptable echo delay.

IV. Methodology

In this section we will present our proposed model comprised of two independent loss models that are based on the transport layer protocol identified in the packets, TCP and UDP. Note that a delay-based approach is ineffective for our purposes since only delays of longer than 600 ms degrade VoIP applications noticeably [20]. Clearly, such long delays violate the SSH QoS requirement of within 250 ms echo replies.

A. UDP Model

Since most VoIP applications use the UDP-based RTP, we expect throttling UDP to degrade them. Only bursty losses (not single losses) effectively degrade VoIP applications. To introduce random bursty losses to the packet flows we use a modified Gilbert Model. The Gilbert model [21] is a two-state first-order Markov chain model that describes typical Internet losses. In our altered model, each stream (identified by its source and destination addresses and ports) is in either the *normal* or *loss* state at any given moment. As Fig. 1 shows, upon arrival of a packet from a fresh stream, with probability p, the stream enters the loss state, causing that packet and the next $N-1$ packets to be dropped. Otherwise, the packet passes and the flow is in the normal state. We apply this procedure to all future packets in the stream. Streams in the loss state return to the normal state in one of two ways: (1) if N unique consecutive packets are dropped, or (2) the timer T expires. The number N ensures bursty losses, and the timer T guarantees that no stream stays in the loss state indefinitely.

We must choose the smallest values for parameters N, T, and p that successfully degrade the VoIP application to minimize the negative effect on other applications. We chose $N = 4$ as the smallest possible value for the number of consecutive packets that VoIP applications cannot tolerate. Similarly for T, we chose 3 seconds, which is short but sufficiently long enough to hurt VoIP applications. Based on our subjective testing (Section V), $p = 10\%$ degrades voice quality sufficiently.

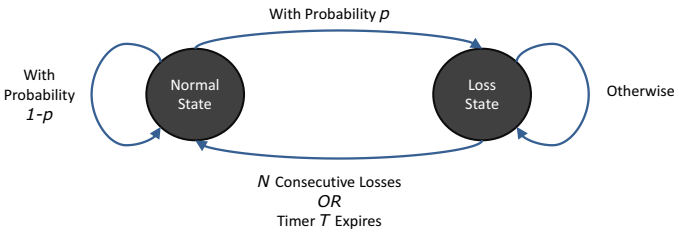

Fig. 1. UDP Model

Since among our permitted applications only DNS uses UDP, and since DNS QoS must be maximized, we let all DNS packets go untouched. We require that all DNS requests are resolved by local DNS servers, whose values could be configured into our tool. If these servers are sufficiently compromised to allow tunneling over DNS, the network owner has far worse problems than the undesirable use of VoIP.

B. TCP Model

Some VoIP applications use TCP if UDP doesn't work. However, throttling TCP traffic would seriously damage permitted applications. As Table I shows, even applying UDP model values of p as low as 1% to TCP severely degrades FTP's performance. This is due to TCP's congestion control mechanism's response to bursty losses [22].

TABLE I
Effect of UDP design on FTP (RTT=200 ms, File size=700MB)

p %	Expected Delay (min)	Accepted Delay (min)	Actual Delay (min)
1	9	27	61
5	9	27	87
35	9	27	112

We therefore used a model that switches between two states: the throttling state where all packets are dropped, and the normal state, with no dropping for a fixed interval. These *normal intervals* permit the TCP congestion control to adjust to the real network conditions, allowing data transmission to take place at full speed during these intervals. Obviously, the normal intervals must be short enough to prevent any satisfactory voice conversation to take place, suggesting a length of no more than a few seconds. In *throttling intervals*, we let TCP control packets (i.e., SYN/SYN-ACK/FIN/FIN-ACK) to pass freely, primarily because their loss has a major negative impact on TCP-based applications.

To improve web use, every stream starts with a longer state, called *initial normal state*, to allow undisrupted download of webpages by avoiding throttling states completely. In Section V, we show that 15 seconds is a suitable value for this interval in our network environment.

A low throttling interval could lead to serial timeout, which TCP congestion control reacts to so radically that even when it is back to the normal state, very little data is transmitted for up to one minute [22], leading to very low utilization. Therefore, we must choke the TCP sender when the stream is in the throttling state, and allow the sender to transmit data at full speed when back in normal state by manipulating the TCP sender's congestion window. Many solutions have addressed a similar problem of temporary disconnections in a mobile environment. We adopted a modified Freeze-TCP [23], because it does not require any changes to either the sender or receiver. The changes are only to what we call the *Dissuade router*, making it possible to fully inter-operate with the existing infrastructure. Freeze-TCP is based on the fact that the congestion window size (thus the sending rate) is determined as the minimum of the receiver's advertised buffer space and the perceived network congestion. If the receiver advertises a zero buffer space, the sender should not transmit data until the client advertises availability in its buffer space [22], and thus no loss is detected by the congestion control system. With similar behavior, our Dissuade router, before entering the throttling state, advertises a zero receiver window size on behalf of the receiver by sending zero window advertisements (ZWAs). ZWAs are acknowledgments of the last data received with the receiver's window being zero. The TCP sender enters the *persist mode* and keeps relevant TCP variables, including congestion window, frozen. When it returns to the normal state, our Dissuade router sends three copies of the ACK (TR-ACKs) for the last data segment it received prior to the disconnection. This mechanism allows the TCP sender to maintain timer values as well as the congestion window without the need for any reduction. During the throttling intervals, we send ZWAs periodically (every 10 ms) to keep the sender in the persist mode. If the sender dishonors the ZWAs, he will experience poor performance for both permitted applications and the targeted application (VoIP).

The *warning period* is the short period prior to the disconnection when our Dissuade router begins sending ZWAs. This period should be long enough to ensure that the sender receives the notification before any packets are dropped, but short enough to avoid forcing the sender into persist mode prematurely. RTT seems a reasonable warning period.

Our model must also ensure that interactive shell applications will work even in throttling intervals. We use a *quota*, the allowed bandwidth during the throttling interval times the length of the throttling interval. Once the quota is reached, all subsequent packets are dropped until the stream enters the normal state. The quota must be large enough to allow interactive shell applications to work well but low enough that VoIP applications use up the quota well before the end of the throttling interval. In our experiments, we picked this quota to be 10 kbps. Skype, for instance, requires at least 32 kbps to maintain a reasonable call quality [14].

When the normal interval ends, the stream enters the *quota state*. Here, every packet goes freely, but we track the bandwidth used. If the quota is used up before the end of throttling interval, the stream enters the *warning state*, where the Dissuade router attempts to persuade the sender to freeze its TCP congestion control parameters using ZWAs. The warning state ends after exactly RTT, when the stream enters the silence period, where all further packets are dropped. In the silence period, ZWAs are sent every 10 ms to both sender and receiver until the throttling interval expires. Fig. 2 shows the quota system for both applications that consume a high bandwidth (such as FTP) and those that use less bandwidth (such as SSH).

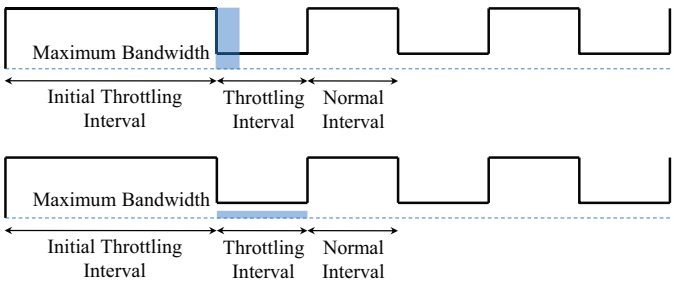

Fig. 2. Illustration of the quota system during the throttling interval for FTP (upper) and SSH (lower)

V. Experiments

We must show that quality of service is achieved for permitted applications but not for VoIP. In this section we demonstrate the degree to which we succeeded by showing experimental results for each application. To perform these experiments, we implemented our Dissuade router using the Click Modular Software Router [24].

A. VoIP

To demonstrate effective dissuasion of VoIP applications, we used subjective testing. While our goal is to throttle the entire class of VoIP applications, we base our voice quality evaluation on Skype. We chose Skype due to its popularity and its aggressive attempts to maintain reasonable quality even under harsh network conditions.

We performed ACR [16] subjective testing on the MOS scale. We have only performed unidirectional testing because of the lack of a standard subjective test that captures all factors in two-way conversations and general difficulties in running consistent two-way tests. We believe that our unidirectional testing results apply to two-way conversations, as well.

Thirty two subjects listened to five 1-2 minute pre-recorded texts, each under a different network condition, and then rated the quality. Both the sender and receiver were in the same local area network, with one behind the Dissuade router. This choice ensured that the parties had low RTT (about 3 ms), the most favorable situation for Skype to cope with a harsh network environment. We believe that higher RTTs do not improve, and may damage, Skype's robustness to difficult network conditions. To test our TCP model, we imposed a high probability loss for UDP traffic to force Skype to switch to TCP. We used 8 seconds for the normal interval and 16

seconds for the throttling interval.[2] We also asked subjects to rate the quality without network problems to measure a baseline for voice quality over unfettered TCP.

We complied with the standards specified in ITU-T Recommendation P.800 for subjective testing [16], including guidelines on format, testing environment specifications, and prerecorded texts. Our results, Fig. 3, show that for both UDP (with loss probability 10%) and TCP models all subjects rated the quality unsatisfactory (i.e. below 3).

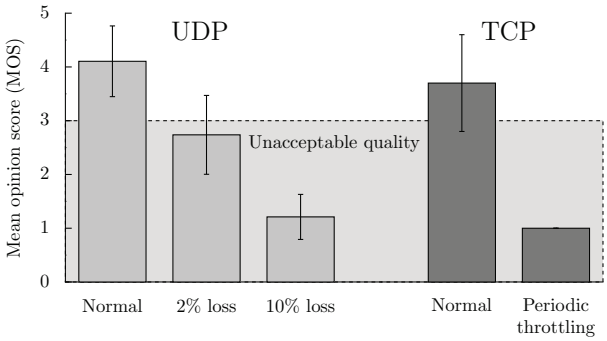

Fig. 3. VoIP subjective testing results

B. Web

To minimize the effect on web usage, the initial normal state must be sufficiently long. We used the top 50 most popular web sites [25] for our experiments. These websites are geographically diverse (spanning multiple continents) and provide distinct contents (including news, social network, search engines, blogs, and entertainments). For each webpage, using the Mozilla Firefox Firebug plug-in, we measured the `index.html` download time and the complete download time—the time from request send until all objects in the webpage (such as images, CSS files, and Java Script files) are completely downloaded. This process was repeated ten times, each at a different time of the day and at a different day of the week to account for Internet variability. As Fig. 4 shows, 47 out of 50 websites were always completely downloaded within 15 seconds, suggesting a suitable value for the initial normal interval in our environment. We used this value in all the experiments presented in this paper.

For the remaining three websites, in a separate experiment, we measured their complete download time behind the Dissuade router. In this experiment, all downloads were completed before reaching the quota in the first throttling state, resulting in no effect from our tool.

C. FTP

FTP's performance is mainly influenced by RTT, available bandwidth, file size, and the TCP congestion control mechanism used. In our experiments, we tested two widely used TCP congestion control variants: Compound TCP with default settings in Windows 7 Professional, and Cubic as it appeared

[2]These values were chosen to maintain a 1 to 2 ratio to meet FTP's QoS (Section III) To dissuade any VoIP conversation, the normal interval should be adequately short. Any pair of values with these characteristics is a potential candidate.

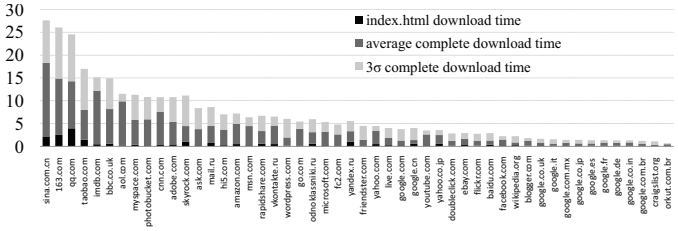

Fig. 4. Web download time analysis (in seconds)

in Linux kernel 2.6.16. For the other variables, we tested values that are expected to lead to the worst FTP performance, to show that even under this condition, FTP's QoS is met. For bandwidth, we used a high bandwidth (100 Mbps) because theoretically TCP takes longer to reach the maximum transmission rate for high bandwidths after loss recovery. For file size, we used a relatively large file (700 MB) to ensure crossing some throttling intervals. We chose a relatively long RTT (200 ms, typical intercontinental distance) since higher RTTs make congestion control take longer to reach the most efficient state [22].

To evaluate FTP, we prepared an isolated and controlled testbed environment to avoid the influence of unknown Internet variables on our results. We ran the experiment ten times independently. As shown in Table II, for both of the congestion control variants, the objective of the QoS not exceeding three times the expected delay was reasonably achieved.

TABLE II
COMPARISON OF AVERAGE DOWNLOAD TIME (IN SECONDS) WITH INSIGNIFICANT STANDARD DEVIATION

TCP Variant	Normal	Dissuade	Accepted Delay
C-TCP	683	2006	2049
Cubic	660	1994	1980

D. SSH

For SSH echo replies should arrive within 250 ms, so the quota should be large enough to allow all SSH traffic to go through. The quota used in our experiments was 20,000 bytes. Recall from Section IV, this number is the product of the length of the throttling interval (16 seconds) and the maximum bandwidth allowed within that interval (10 kbps). To confirm this quota's sufficiency, users' typing behavior was simulated by generating 10,000 characters, each for some fraction of a second. According to Ostrach [26], the top 10% fastest typists type at a rate within 65-105 words-per-minute (WPM). We simulated users' typing behavior within this range, confirming that even for the fastest typists (WPM = 105), the quota is never reached (total used quota = 13,254 bytes).

VI. DISCUSSION

A. Did Our Chosen Experiment Work?

We degraded VoIP to a degree that all our human subjects agreed was unsatisfactory, while we met the target metrics for success at the other applications. However, these target metrics are debatable. The "acceptable" delay for web browsing is arguably unacceptable, or much longer delays might be

acceptable, which may be even truer for file transfer delay. Further, what about other applications, like on-line gaming or telemedicine? We achieved success within the bounds of our experiment, but cannot say what happens with a slight expansion of those bounds.

Another question is whether we stopped VoIP regardless of what countermeasures the VoIP system designers might take. While we did account for some obvious countermeasures, there may be other countermeasures that would neutralize network dissuasion. For instance, the clever division of a VoIP flow into several distinct flows might result in the VoIP flow slipping by.

In summary, at best we achieved successful dissuasion under a particular well-defined set of conditions, including certain questionable assumptions about attacker behavior.

B. Did We Validate the Concept of Network Dissuasion?

Achieving success for one set of conditions does not imply that network dissuasion is generally useful, or that dissuasion models can be readily generated. We thought shutting down VoIP would be easy. It wasn't. It required multiple model enhancements, and our final model required periodic behavior switching, specific startup behaviors, threshold levels, and special cases for some types of packets. This model is complex and customized for both the dissuaded application and the desirable applications. It may be difficult to design network dissuasion for arbitrarily chosen sets of desirable and undesirable uses.

Further, at best dissuasion can handle classes of uses. It's hard to see how the technique could be used to prevent downloading copyrighted media, while allowing legal downloads of software updates, for example.

Perhaps if we had a deeper understanding of applications' network behaviors, we could identify matches between network characteristics and application requirements that would allow easier creation of customized dissuasion parameters. Thus, the concept is a reasonable area for further investigation.

VII. Concluding Remarks

We demonstrated that network dissuasion is a feasible technique for one particular set of non-trivial goals. Our experiments showed our ability to make voice over the network intolerable, while simultaneously supporting file transfer, web browsing, and interactive text applications.

However, the difficulty of achieving these goals and the specificity of the methods required cast doubt on whether the technique will be generally useful. The controls were highly tailored to the specific applications. Because we used Skype, rather than a UDP-only voice application, we had to contend with the behavior of all FTP applications. Because we included an application that required a small amount of bandwidth with a relatively low delay, we had to add thresholds in the model. The effort required to achieve these goals would not be possible to frequently expand. Thus, while the question of possibility has been answered, the question of practicality has not.

This work highlights our limited understanding of what applications demand from networks. As the range of network applications grows and the use of networks increases, we will find a greater need for understanding application requirements. Perhaps even if network dissuasion is not practical, it will help us develop our understanding of the interactions between network and application behavior.

References

[1] X. Wu, I. Lambadaris, H. Lee, and A. R. Kaye, "A comparative study of some leaky bucket network access schemes," in *Proc. of SUPERCOMM/ICC*, 1994.

[2] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *Trans. on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[3] I. Wakeman, A. Ghosh, J. Crowcroft, V. Jacobson, and S. Floyd, "Implementing real time packet forwarding policies using streams," in *Proc. of USENIX*, 1995.

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Trans. on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," *RFC 2475*, 1998.

[6] Cisco Networks, "Comparing traffic policing and traffic shaping," *Cisco document 19645*, 2005.

[7] P. Syverson, D. Goldschlag, and M. Reed, "Anonymous connections and onion routing," in *IEEE Journal on Sel. Areas in Comm.*, 1998.

[8] M. Shapiro, "Structure and encapsulation in distributed computing systems: the proxy principle," in *The 6th International Conference on Distributed Computing Systems*, 1986.

[9] "HTTP tunnel," http://www.nocrew.org/software/httptunnel.html.

[10] "DNS tunnel," http://dnstunnel.de.

[11] W. Jiang and H. Schulzrinne, "Comparison and optimization of packet loss repair methods on VoIP perceived quality under bursty loss," in *Proc. of NOSSDAV*, 2002.

[12] L. De Cicco, S. Mascolo, and V. Palmisano, "An experimental investigation of the congestion control used by Skype VoIP," *Wired/Wireless Internet Comm.*, pp. 153–164, 2007.

[13] L. De Cicco and S. Mascolo, "A mathematical model of the Skype VoIP congestion control algorithm," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 790–795, 2010.

[14] S. A. Baset and H. Schulzrinne, "An analysis of the Skype peer-to-peer Internet telephony protocol," in *Proc. of INFOCOM*, 2006.

[15] H. Y. Yang, K. H. Lee, and S. J. Ko, "Communication quality of voice over TCP used for firewall traversal," in *Proc. of ICME*, 2008.

[16] ITU-T, "Methods for subjective determination of transmission quality," *Recommendation P.800*, 1996.

[17] Nortel Networks, "QoS performance requirements for UMTS," 1999.

[18] A. Bouch, A. Kuchinsky, and N. Bhatti, "Quality is in the eye of the beholder: meeting users' requirements for Internet quality of service," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2000, pp. 297–304.

[19] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W. M. Yao, and S. Schwab, "A user-centric metric for denial-of-service measurement," in *Proc. of Workshop on Experimental Comp. Sci.*, 2007.

[20] P. T. Brady, "Effects of transmission delay on conversational behavior on echo-free telephone circuits," *Bell Syst. Tech. J*, vol. 50, no. 1, pp. 115–134, 1971.

[21] E. N. Gilbert and Others, "Capacity of a burst-noise channel," *Bell Syst. Tech. J*, vol. 39, no. 9, pp. 1253–1265, 1960.

[22] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-host congestion control for TCP," *Comm. Surveys Tutorials*, vol. 12, no. 3, pp. 304–342, 2010.

[23] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *Proc. of INFOCOM*, 2000.

[24] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *Trans. on Comp. Systems*, vol. 18, no. 3, pp. 263–297, 2000.

[25] "Most Popular Websites on the Internet," http://mostpopularwebsites.net.

[26] T. R. Ostrach, "Typing speed: How fast is average?"