DeltaShaper – Video Stream Synthesis for Internet Censorship Circumvention

Diogo Miguel Barrinha Barradas
diogo.barradas@tecnico.ulisboa.pt

Instituto Superior Técnico
(Advisors: Professors Luís Rodrigues and Nuno Santos)

Abstract. Citizens living in countries ruled by repressive regimes may have access to rightful information blocked or may be prevented from sharing such information freely on the Internet. In spite of all efforts to hinder the transmission of potentially compromising data, even the most oppressive regimes cannot afford to block all channels with the outside world, as these may be instrumental to preserve the economy and the sustainability of the regime itself. One way to circumvent censorship is to use those channels that cannot be blocked as carriers for *covert channels*. In this project we study techniques to hide arbitrary sequences of data in the traffic generated by popular multimedia streaming applications, such as Skype and Google Hangouts, since these applications are widely used for business and private interaction and therefore cannot be blocked without significant social and economic impact to the repressive regime. Although tunneling protocols with this characteristic have been proposed in the past, they are limited on the type of information that can be transferred and on the type of interference they tolerate. We present a novel technique to encode a covert channel in synthesized video frames which can be transmitted through a video streaming protocol.

1 Introduction

Today, most electronic communication over the Internet can be controlled by the government and/or by a few corporate players. Repressive regimes may monitor and control the access to the Internet by employing several censorship techniques, such as blocking specific IP addresses or specific content (for instance, web sites). As a result, citizens may see their rights to freely access rightful information, communicate, and express opinions severely constrained [1–3].

However, even the most oppressive regimes cannot afford to block all channels with the outside world, as these may be instrumental to preserve the economy and the sustainability of the regime itself. In particular, there is evidence that several countries do restrict access to information but maintain operational widely used services such as Skype. In fact, only in extreme scenarios, governments can afford to completely block the access to the Internet [4, 5].

In certain cases, to prevent access to specific sources of information, the censor employs simple blacklisting techniques in which it instructs ISPs to block direct connections from the population to these sources. To circumvent such restrictions, a typical strategy consists in accessing this information via a trusted proxy. However, this strategy is only viable as long as the proxy address is not public and the connections to it cannot be easily flagged as suspicious.

In order to prevent all proxies addresses to become public, Tor [6] has suggested the use of *bridges*, proxies which addresses are not publicly distributed, to access its overlay network. Even so, if no explicit effort is made to obfuscate the traffic towards a bridge, it may exhibit patterns that make it easily recognizable [7]. Unfortunately, the task of obfuscating the traffic is a challenge by itself. If the resulting pattern does not match any known protocol, the flow can also be deemed as suspicious. Therefore, for the obfuscation to succeed, the resulting traffic should mimic existing protocols, ideally protocols that a censor may not be willing to block. But protocol mimicking can be extremely hard to implement [8] and needs to be adjusted every time the protocol is updated.

A more recent and promising approach consists in providing access to rightful information using a covert channel by *tunneling* the data stealthily through protocols that are unlikely to be blocked by the censor. Here, the assumption is that it is possible to implement the tunneling in such a way that the traffic pattern of the carrier protocol is not modified in a significant way. Examples of this line of work include FreeWave [9], which encodes network traffic into acoustic signals sent over VoIP connections, and Facet [10], which enables clients to secretly stream censored videos over a Variable Bit Rate (VBR) video conferencing call. To assure "unobservability", Facet embeds the censored video into some carrier video such that the transmission of both the resulting video and the streaming of an unsuspected video cannot be distinguished by a censor agent that can listen to the network and perform traffic analysis or active packet manipulation.

These results are very promising but still have important limitations. FreeWave is vulnerable to attacks that leverage perturbations in the network. The drop of selected packets may render the system useless, as it will corrupt the negotiation of parameters for the audio data demodulation. Also, its covert channel can be uncovered when established over a VBR VoIP connection, by analyzing the generated network traces. In its turn, Facet employs a technique named *video morphing* which can only be used to transfer video content. This restriction may severely limit Facet's broader applicability to other important types of communication, namely web browsing and bulk file transfers.

To bridge this gap and support covert transmission of arbitrary sequences of data frames, a naive approach is to extend Facet's pipeline with additional processing stages. At the sender side, a data frame must first be encoded as a "payload video" before it can be embedded into the carrier video and transmitted over the wire. Also, a new last mile stage must be added at the receiver side in order to decode the extracted "payload video" and obtain the original data frame. Although this approach may seem conceptually simple, data encoding/decoding raises non-trivial challenges. The first challenge is about space efficiency. To avoid wastage of covert channel bandwidth, which is important for bulk data transfers, we should be able to fit as many bits as possible in each of the frames of the payload-carrier video. The second challenge is about time efficiency. To prevent significant data delivery latency, which is important for interactive communication such as web traffic, encoding and decoding must add small delays to the communication. A third challenge is to achieve space and time efficiency without degrading a system's "unobservability" property.

In this work, we aim at mitigating the limitations imposed by existing systems. We design a novel data encoding technique that allows the transmission of arbitrary sequences of data frames through a synthesized video stream over a cover protocol.

The rest of the report is organized as follows. Section 2 briefly summarizes the goals and expected results of our work. In Section 3 we present an overview of the censorship circumvention techniques described in the literature. Section 4 details several protocol tunneling approaches, as implemented by existing systems. Section 5 describes the proposed architecture to be implemented, and Section 6 describes how we plan to evaluate our results. Finally, Section 7 presents the schedule of future work, and Section 8 concludes the report.

2 Goals

This project aims at exploring approaches that allow for censored information to be accessed via a covert channel that is tunneled through protocols which are hard to block by a censor. Although previous work has started to explore this approach, existing systems still exhibit significant limitations, as presented in the introduction. Our goal is to mitigate some of these limitations.

> *Goals:* This project focuses on the design and implementation of a new censorship circumvention system that introduces a novel data encoding technique which allows for the encoding of a reliable data stream within a synthesized video stream. The synthesized video stream shall exhibit the traffic profile of a regular video call, such that the covert channel cannot be identified via traffic analysis tools. We expect to be able to achieve these goals while sustaining a latency that is sufficient for interactive communication, such as web browsing, and a throughput that can accommodate bulk data transfers, such as large documents.

In order to evaluate the proposed solution we will implement a prototype of the system that will use a Skype's call video stream as the carrier for the covert channel. An extensive experimental evaluation of the system will be performed in order to assess: the resilience of the covert channel to attacks on the traffic flow that do not compromise a regular Skype call; the resilience to traffic analysis, by comparing the resulting flows with flows from unaltered Skype calls; the maximum throughput that can be achieved; and the minimum latency that can be attained. In summary, the project will produce the following expected results:

> *Expected results:* The work will produce i) a specification of the system; ii) an implementation adapted for Skype video calls, iii) an extensive

experimental evaluation of the performance and resilience of the system to active and passive attacks.

3 An Overview of Censorship Circumvention Techniques

In this section we give an overview of the main Internet censorship circumvention techniques that have been described in the literature. Naturally, the likelihood of success of these techniques depends on the ability of the censor to detect and interfere with the information flow it is trying to block; the more powerful the censor is, the more sophisticated the circumvention technique has to be. Therefore, we start by briefly discussing the power of the censor and the attacks that the censor can perform in the information flows. Then we proceed to make a survey of the main circumvention techniques in increasing order of sophistication, with particular emphasis on techniques that leverage the fact that some protocols/services cannot be easily blocked by the censor.

3.1 Power of the Censor

The ability of a censor to efficiently detect and/or block some target information flows depends on the amount of resources it has available but also on its know-how and proficiency. In our survey we consider the classification proposed by Houmansadr et al. [8]. Three levels, which take into account the resources available to the censor, are proposed: *local adversary*, *state-level oblivious adversary* and *state-level omniscient adversary*. A local adversary (such as a corporation) is only able to observe a limited number of connections. On the contrary, state-level adversaries can observe connections on a larger scale, often with state-sponsored ISPs cooperation. A state-level oblivious adversary is expected to perform Deep Packet Inspection (DPI) only to short observations of network traffic, considering that its infrastructure does not allow to keep connection records in long-term. Instead, a state-level omniscient adversary has the required infrastructure to collect, store and analyze a large set of network traces.

In this work, we face a state-level omniscient adversary, which is assumed to be able to observe and store all the network flows crossing its borders, but unable to snoop the digital contents in end-users computers. It is also unable to control the software installed on end-users computers. In fact, government initiatives directed at increasing the reach of Internet censorship to the edges of the network, such as China's Green Dam [11], have failed in the past.

3.2 The Censor Toolbox

Broadly speaking, there are three major categories of attacks that a censor may perform to detect or to prevent the access to rightful information:

– *Passive attacks* are tied to the analysis of observed and collected network traces. Typical passive attacks are conducted with statistical traffic analysis resorting to DPI mechanisms, usually deployed within ISPs' premises.

– *Proactive attacks* are performed by sending crafted probes to random or suspected hosts in order to get a response that may identify them as circumvention mechanisms. Usually, the censor acts as a client of the censorship circumvention system itself.
– *Active attacks* are perpetrated by perturbing network traffic. A censor may be able to delay, drop, modify or inject content into selected network packets.

To perform these attacks the censor may employ a large spectrum of techniques aimed at identifying, interfering with, or blocking information flows. Such techniques range from simple IP address blocking to actual monitoring and manipulation of served content. DPI allows for the gain of information about the length and content of packets that cross a network. This enables a censor either to take immediate action based on the content of each observed packet or to employ statistical traffic analysis techniques, which usually require the ability to store a large number of network traces [12]. Such statistical analysis may be able to unveil unusual network traffic patterns that may be linked to the use of censorship circumvention mechanisms.

3.3 Censorship Arms Race

Censorship circumvention systems are continuously at an arms race against increasingly sophisticated censorship techniques. Any circumvention technique aims at achieving both "unobservability" and "unblockability":

– A censorship circumvention system lacks "unobservability" if it can be identified by the censor. To achieve "unobservability", circumvention techniques typically rely on the existence of a pre-shared secret [13–16] and/or on the the ability to obfuscate the communication [17–19]. These two approaches can make it harder for the censor to detect covert communications in the network (among all other innocuous traffic).
– A circumvention system is said to be "unblockable" if the flows it protects cannot be blocked without substantial loss for the censor. One of the most promising strategies to achieve this property is to tunnel flows through widely used protocols such as cloud services, email, VoIP or video conferencing [20, 21, 9, 10], in such a way that the only remaining strategy for the censor is to apply indiscriminate attacks to all flows, even to those that the censor would like to preserve.

Clearly, a system that is observable becomes more vulnerable to be blocked, given that the censor is then able to stage a targeted attack instead of an indiscriminate attack. It is therefore no surprise that achieving "unobservability" is referred by Houmansadr et al. [9] as the biggest challenge facing the existing censorship circumvention systems.

In the following paragraphs, we present an evolution of censorship-enforcing techniques as well as systems designed to circumvent them.

3.3.1 Content Withholding

Early state-sponsored censorship techniques relied on the ability of the censor to control publication mediums, such as newspapers or TV shows. Before being published, all content would be previously scrutinized, to ensure that harmful information to the regime would never be divulged.

Recently, the Internet has become an excellent medium to exchange information and therefore an obvious target for censors. Although many Internet services are neither hosted nor controlled by repressive regimes, they may still offer censorship-enforcing platforms to prevent offensive content from being distributed within a country's borders.

Techniques available to the censor: Some popular Internet services, such as Twitter, enable governments to formally request the withholding of content within a country's boundaries [22]. This is mostly due to business decisions, enabling the service to remain available within several countries, even those whose governments actively perform Internet censorship.

Circumvention strategies: In order to stealthily transmit information that would otherwise be censored, traditional steganographic techniques aimed at concealing the existence of a message by embedding it into some cover medium [23]. Examples of such techniques are the concealment of Morse code in the length of grass in a drawing or Microdots, which allow for the representation of printed material in small dots. Variations of these steganographic techniques are still employed in the digital age, where data may be encoded within different layers, such as text or media files [24, 25]. Such techniques may be used by citizens living in countries ruled by oppressive regimes, in order to publish and access rightful information, by evading simple content filtering mechanisms and not having their messages targeted by content withholding requests. This may force the censor to effectively block the access to the Internet service itself, as it may not be able to distinguish innocuous from steganographically-marked communication.

3.3.2 Simple Internet Destination Blocking

Destination blocking is one of the simplest strategies that can be used by a censor to prevent users from accessing rightful information in the Internet. It consists of identifying the endpoints that serve the content to be censored and then block the access to those endpoints. In many cases, this strategy can be implemented even by a local adversary possessing few computational resources.

Techniques available to the censor: Consider the case where the information sources can be identified by an IP address. In this case, a censor may choose to enforce either a blacklist or a whitelist of IP addresses [1]. While being more permissive, a blacklist is easier to manage than a whitelist, considering that in the latter each new host on the Internet would need to require an authorization from the censor in order to become accessible within its network.

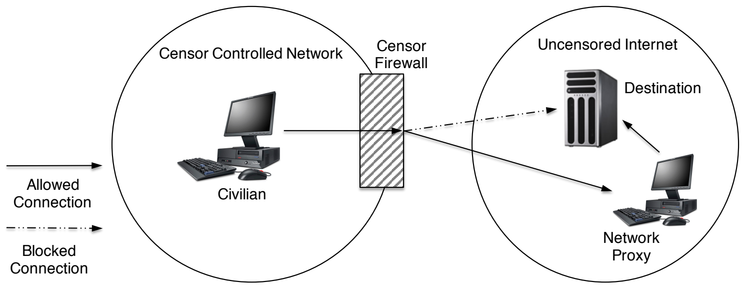

Fig. 1. Proxy-based censorship circumvention overview.

The interception of DNS Lookup requests can also be used by the censor in order to prevent the translation of a given hostname to its IP address. The censor responds with an IP address serving a different page, typically under its control, rather than the originally requested one [1].

A censor may also block connections in a more selective fashion. If the content is served over HTTP, the censor may filter connections based on the headers of HTTP requests. This kind of attack allows for blocking content at a finer granularity, restricting access only to certain pages on a given domain [1]. This is performed by analyzing a page's full URL, looking for specific keywords.

Circumvention strategies: The described censorship techniques can be evaded by leveraging proxy-based traffic re-routing, which is often combined with digital steganography techniques.

Instead of directly establishing a connection with the host serving blocked content, a user could direct its request to a network proxy, which would instead be accessible, enabling seemingly uncensored web-browsing. This is depicted in Figure 1 and implemented by systems such as Ultrasurf [26].

Decoy routing systems [15, 16, 27] aim to deploy special routers within co-operating ISPs' networks. The key insight of this technique is to have routers, rather than end hosts, relay traffic to blocked destinations. A client issues an HTTPS steganographically marked request to an overt destination whose path crosses a decoy router deployed by an ISP. Decoy routers are able to recognize such mark (which reveals a client's true desired destination) and act as a man-in-the-middle, diverting traffic to blocked destinations. The censor is only able to observe the innocent looking destination in HTTPS requests.

A different approach employing steganography relies on volunteer relay servers which secretly serve censored content. A client wishing to use Infranet [13] places visible, seemingly innocuous HTTP requests with associated additional semantics, regarding the sequence of requests performed. Infranet servers interpret such sequence, steganographically embedding the requested content into uncensored images that are returned to the client. We must highlight that Infranet's threat model consider the blocking of HTTPS connections to be a reality.

3.3.3 Network Proxies Detection and Blocking

As already discussed, the use of network proxies may be leveraged to evade censorship techniques that rely on the identification of the source of information. However, it is often difficult to hide the proxies from the censor and, once proxies are detected, they can be blocked in a similar manner as the original source.

Techniques available to the censor: A censor may be able to block connections to a given source of information, even when users re-route their traffic through network proxies. By gathering the addresses of publicly distributed proxies, a censor is able to blacklist connections to such hosts.

A censor may also be able to detect connections to unlisted network proxies. If a client connects to a network proxy through HTTP, the censor may be able to inspect the network packets' contents and flag the connection for prohibited content. Not only that, the censor may actually be able to modify a packet's content itself, preventing prohibited information to be delivered. The same strategies can be used to detect an Infranet server. If the website serving the covert content does not have a legitimate reason to change its cover content frequently, a censor may notice that the same cover content structure appears to be different when embedding different covert data.

Schuchard et al. proposed an attack [28] where censors may proactively probe and detect routes containing decoy routers. Thus, censors capable of making routing decisions could avoid to send traffic through such routes. However, such attack was later studied by Houmansadr et al. [29], who argue that a strategic placement of decoy routers would render it ineffective.

Providing that HTTPS is allowed and the client uses an HTTPS proxy to circumvent censorship, a censor may employ known website fingerprinting [30] techniques to look for statistical deviations of the censor's regulated traffic, identifying ciphered connections that are being used to circumvent content filtering. The same applies for connections between a client and decoy routers, where the traffic fingerprint of the covertly requested website does not match the one of the seemingly requested website.

To flag the aforementioned HTTP connections, a censor must at least be able to perform line-speed DPI, in order to analyze the properties of single packets flowing through the network. This type of censor falls into the category of state-level oblivious adversary. Conversely, website fingerprinting techniques require the analysis of large data sets of network traces, so that they can yield more accurate results. If a censor drops connections while having a high degree of uncertainty about their nature, it is likely that regular connections will be affected. Therefore, only a state-level omniscient adversary owns this capability.

Circumvention strategies: A censor may still be able to detect connections to unlisted proxies through a connection's characteristics and content. A new class of systems helping in circumvention aim to obfuscate connections' traffic, transforming it in such a way that it cannot be linked to the underlying application layer protocol. Such technique is dubbed *traffic morphing* [31]. One of such obfuscation methods consists in *protocol randomization*, where all the traffic from a regular connection is ciphered to make it seem random from a network's observer point of view.

An example of an overlay network that uses a proxy approach in order to support censorship circumvention is Tor [6]. In Tor, such proxies are called bridges, representing hosts whose IP address is not publicly advertised. Bridges can be discovered through passive attacks by observing Tor connections' characteristic traffic, which uses TLS in its outer layer. Such attacks may require line-speed DPI [7] or the storage and posterior analysis of network flows [32, 33].

In order to avoid the fingerprinting of Tor traffic at line-speed [7], Obfsproxy [34] uses a stream cipher to encrypt regular Tor traffic. This hides the TLS cipher suite list, server name and TLS extensions, which could give the connection away, by making the traffic effectively look like random bytes. However, the use of length-preserving encryption does not prevent detection from other traffic analysis techniques, which take into account the distribution of packets' length and inter-arrival times. To protect against this kind of attack and to enhance the randomization mechanism, ScrambleSuit [35] is used in tandem with Obfsproxy, manipulating packets' length and inter-arrival times.

The above protocol randomization techniques were devised to work alongside Tor as modules of an obfuscation framework named pluggable transports [36]. These obfuscation systems assure that simple protocol classifiers will be unable to identify the morphed protocol which, in this case, appears to be random.

3.3.4 Protocol Randomization Detection and Blocking

While protocol randomization can evade detection from simple protocol classifiers, a censor may take a different approach, whitelisting approved protocols and throttling the connection for unknown or undesirable protocols.

Techniques available to the censor: While protocol randomization systems are able to provide a fast and efficient traffic morphing, the network traffic they generate is distinguishable as it does not resemble any known protocol. Therefore, they are defeated by a censor which only authorizes known and approved protocols across its network.

It is possible to distinguish connections performed through protocol randomization from regular TLS connections. While Obfsproxy encrypts every flow message, TLS uses fixed plaintext headers when establishing a connection, as part of the TLS handshake [37]. Entropy tests on initial messages' headers may be able to distinguish regular TLS traffic from Obfsproxy encrypted traffic. Concretely, such entropy tests may indicate a uniform byte distribution where the plaintext TLS headers should be located [38]. This analysis does not require comparison and correlation with large data sets of stored packet samples, enabling a state-level oblivious adversary to block an unknown or undesirable protocol.

Circumvention strategies: A different strategy for protocol obfuscation is *protocol imitation*, which aims at mimicking known and popular protocols permitted by a censor, evading the possible throttling of randomly morphed ones. The advantage lies in the unwillingness of a censor to indiscriminately block popular and fundamental services to the region's economical development. As such, existing systems try to mimic protocols like HTTP and several VoIP implementations.

To avoid traffic analysis of a connection's packets distribution, StegoTorus [18] steganographically conceals chops of Tor traffic on a cover protocol messages. The chopped traffic is split over multiple connections and can be delivered out of order. Each connection is served by a steganography module, which may act like an HTTP or VoIP client/server.

SkypeMorph [17] was designed to obfuscate connections to Tor bridge nodes by mimicking the statistical properties of Skype's video calls. Unlike StegoTorus, a SkypeMorph client initiates a connection by effectively calling a bridge, quickly dropping the call thereafter. However, the call is seemingly uninterrupted as the communication with the bridge continues by exchanging the morphed network packets carrying Tor's traffic.

CensorSpoofer [19] leverages a similar approach to both Stegotorus and SkypeMorph, while attempting to keep its server location hidden from the censor. A client can communicate to the CensorSpoofer server the page it wishes to visit through a low-bandwidth channel, such as a steganographically marked email. The server fetches the requested web page and sends it back to the client by embedding it into the network packets of a spoofed VoIP session imitation.

A different approach at protocol imitation is to use Format-Transforming Encryption (FTE) [39] in order to produce ciphertexts that are able to match regular expressions from a user's choosing. FTE can then be used to foil regex-based DPI systems by producing ciphertexts that match the content definition of a protocol allowed across the censor's network.

Protocol imitation obfuscation techniques are still able to defeat the early discussed traffic analysis techniques based on the network packets' length and inter-arrival times distributions. Moreover, they are able to evade blacklisting by mimicking popular and allowed protocols within the censor's controlled network.

3.3.5 Protocol Imitation Detection and Blocking

Houmansadr et al. [8] consider that achieving "unobservability" through protocol imitation presents a hard challenge. This is because a system must completely mimic the target protocol behavior, including error conditions and implementation specific bugs. Additionally, protocols may show dynamic dependencies among several established connections, adding to the complexity of a correct imitation.

Techniques available to the censor: To flag or thwart a covert channel over protocol imitation only requires the censor to spot a single discrepancy when comparing with the real protocol. Favoring the censor, a large part of good candidates for imitation are also complex protocols with many of the aforementioned dynamic dependencies. Thus, even a local adversary may be able to distinguish between a legitimate protocol and an imitation by performing active or proactive attacks. Other imitation techniques may only be detected by leveraging DPI mechanisms that are feasible to be implemented even by low resourceful censors. We present below several attacks that a censor may apply to the previously described protocol imitation systems.

A censor may pro-actively probe StegoTorus servers' HTTP module with HTTP requests, so as to determine if the servers' responses to both correct and malformed requests is consistent with a real HTTP server's software fingerprint. The StegoTorus server may be flagged by a censor, shall it fail to exhibit a usual fingerprint found among real HTTP servers.

SkypeMorph lacks the correct implementation of an accompanying TCP control channel which directly reflects perturbations in the UDP stream and vice-versa. A censor may perform an active attack and then check that such dynamic dependence is not enforced as in a legitimate Skype call.

A censor is able to flag a connection to a CensorSpoofer server by detecting inconsistencies between the actual and the spoofed server. The SIP connection between a client and a CensorSpoofer server is relayed through a registrar outside the censor borders. As such, the censor is unable to verify the server's true IP address. However, a censor may send a SIP INVITE request to a dummy SIP ID on the server's alleged IP address. While a genuine SIP client would return a status message, the spoofed server may not have a running SIP client, therefore not responding to the censor's probe.

The use of FTE may be detected at line-speed by applying entropy tests on the first message of an FTE flow, an HTTP GET request. These yield results with relatively low false-positive rates [38], allowing a censor to flag the connection.

Circumvention strategies: A recent system, Marionette [40], tackles previous mimicking limitations by constructing a hierarchical composition of probabilistic automata, capable of controlling several aspects of protocol mimicry. Automata composition is used to control fine-grained aspects of mimicry, such as dynamic dependencies over channels, statistical properties of generated traffic and error conditions. For instance, it is possible to fully model an HTTP server specification so that a server leveraging protocol imitation resists active probing attacks.

Unlike the previously described imitation systems, Marionette is fully programmable through a domain-specific language, making it easy to adjust the obfuscation strategy, ranging from randomized obfuscation to full protocol imitation, while avoiding the need to rewrite the system from the ground up. However, good candidates for imitation may be proprietary software, demanding its reverse engineering in order to build a model for imitation. Not only this is a tedious effort, it may involve its repetition once a new version of the software is made available.

To rise the difficulty of detection by a censor, *protocol tunneling* systems attempt to evade censorship by tunneling blocked content through an implementation of a protocol. Such as protocol imitation systems, these rely on the unwillingness of a censor to block popular protocols. However, systems that work by tunneling take advantage of directly coping with the chosen implementation intrinsic characteristics. As an example, FreeWave [9] takes advantage of VoIP channels to tunnel modulated acoustic signals which encode data.

3.3.6 Protocol Tunneling Detection and Blocking

Protocol tunneling systems avoid the need to faithfully implement a popular cover protocol, by effectively using it to tunnel covert traffic. While the possible thwarting of covert channels may be performed through active attacks, the censor may unwillingly affect legitimate connections. To avoid this issue, a better approach would be to pinpoint an abusing connection, perturbing it after some degree of certainty it is being used to evade censorship.

Techniques available to the censor: In order to detect a covert channel over a popular protocol, a censor may look for mismatches between the covert and carrier protocols [41]. We refer to the following types of mismatch:

- *Architectural mismatches* may allow the identification of information flows when the cover protocol communication architecture differs from that of the covert protocol. For instance, if protocol tunneling is performed through a centralized system, the server relay is consistent with a client-proxy architecture. Decentralized architectures may turn the endpoint acting as a proxy into a hot spot, raising suspicion from the censor.
- *Content mismatches* may allow for the detection of covert information flows, by observing that the traffic patterns generated by a given protocol are different than expected when it is used to tunnel the covert protocol data.
- *Channel mismatches* may pose a risk for the correct functioning of the covert protocol. A covert protocol which needs reliable transmission may be affected by the design of its cover protocol, which may tolerate the loss, delay or duplication of packets.
- *Utilization mismatches* may set off a response from the censor. Since a protocol not directly intended to browse blocked web content may be used for tunneling, the user of such system must avoid seemingly abusive uses so that circumvention can remain undetectable. As an example, a long web-browsing session over FreeWave may take longer than the average Skype call time [42]. Abnormal connection times through the cover protocol may look suspicious and trigger further investigation from the censor.

The detection of covert channels over carrier protocols requires a passive attack to be performed. So as to detect content or utilization mismatches, the censor must be able to keep a record of past network traces. For instance, connections from users must be recorded in order to establish common utilization rates, such as the average duration of VoIP calls. Traffic analysis aiming to find content mismatches also requires a large data set of network flows to be available. As is, a more accurate detection of a covert channel is in the hands of a state-level omniscient adversary. However, even a local adversary may be able to thwart a covert channel while incurring in minor performance issues for legitimate connections, shall it be able to exploit a channel mismatch between the carrier and covert protocols.

Circumvention strategies: To prevent a local adversary from thwarting covert protocols, protocol tunneling systems should be implemented in such a way they can cope with the chosen cover protocol intricacies. These systems pose as a good approach to censorship circumvention as they may be able to effectively evade simple or line-speed censorship mechanisms, forcing the censor to devise advanced censorship techniques based on traffic analysis. A more detailed description of such systems is presented in Section 4.

4 Protocol Tunneling Revisited

Protocol tunneling surges as one of the best approaches to censorship circumvention. Providing that a protocol tunneling implementation avoids mismatches between the covert and cover protocol, not even a state-level omniscient adversary may be able to infer discrepancies between a regular execution of the legitimate protocol and one being used for evading censorship. The next paragraphs present a survey of existing protocol tunneling systems, concluding with a brief discussion.

4.1 Protocol Tunneling Using Staged Network Applications

The following systems stand upon widely used protocols which use an oblivious server relay to stage both client requests and covert data prior to delivery. They are presented in increasing order of security guarantees.

4.1.1 SWEET

SWEET [21] leverages email communication to tunnel covert traffic through email messages.

Description: SWEET operates by having the client exchange emails that include the covert traffic with a dedicated server. When it receives an email, the SWEET server is able to extract the covert information, process it (for instance, obtain a requested web page) and produce a reply message that tunnels the response back to the client (for instance, the content of the requested web page). The client may contact the SWEET server directly or, if the destination domain needs to be hidden, it can also be accessed indirectly: emails are sent to an account of a widely used service and this account is regularly accessed by the SWEET server (in this case, the credentials for that account need to be agreed using some out-of-band channel).

The way to tunnel information in the email messages depends on the type of email services that are permitted by the censor. If the client can access directly trusted servers using encrypted emails, the information is simply transferred in the body of the email message. If the client is forced to use an outgoing email server controlled by the censor, steganography techniques must be used to encode the covert information in what appears to be a regular email message.

Advantages: A significant advantage of SWEET is that it uses a service which is unlikely to be completely blocked by a censor, given the importance that email has obtained in the daily life of citizens and business. SWEET is also able to provide communication with a sufficiently small latency for supporting web-browsing.

Limitations: A serious limitation of SWEET is that the traffic patterns imposed by the covert channel may be significantly different from the traffic patterns induced by regular email. For instance, under normal use a client may only exchange a few email messages per day, with relatively long inter-message intervals. On the other hand, while browsing the web using SWEET, the client may deal with many (incoming and outgoing) messages in a short interval. Thus, traffic analysis tools are likely to unveil the existence of the covert channel. A better approach would be to make use of a cover service with more flexible traffic and utilization patterns, such as cloud storage, to tunnel covert data.

4.1.2 CloudTransport

CloudTransport [20] leverages cloud storage services to tunnel covert traffic through storage read and write requests.

Description: CloudTransport operates by having the client exchange information with a server that serves as a bridge to the censored service via an intermediate cloud storage service, as depicted in Figure 2. The client writes a request in the cloud storage. In turn, the bridge periodically polls the cloud storage for new files; it fetches the request, serves it and writes the response back to the storage service. Finally, the client reads the storage service to get the reply. For this scheme to work, the client and the bridge must pre-agree on the storage service to use and on location of the files to be exchanged on the storage namespace.

CloudTransport also aims at helping clients to find bridges and negotiate which storage account to use for exchanging the files. For this purpose, each bridge is required to maintain a dedicated cloud storage account that is only used to rendezvous with potential clients. In turn, a client can setup its own cloud storage account for supporting the cover channel and then share the account credentials with the selected bridge, making use of the rendezvous account of that bridge. Furthermore, CloudTransport suggests that some directory service could be created for bridges to advertise their services.

Advantages: As cloud storage services become more and more popular, it may be hard for a censor to completely block cloud storage without significant loss. Also, as the set of applications that use cloud storage is diverse, many read/write

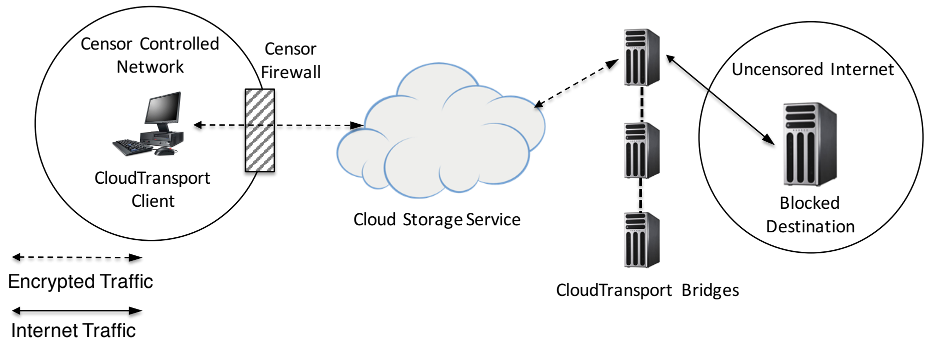

Fig. 2. CloudTransport overview (adapted from CloudTransport [20]).

patterns may exist in practice and this diversity may help in making the covert channel hard to detect.

Limitations: The traffic patterns imposed by the covert channel may still be different from the traffic patterns induced by regular application that use cloud-storage. Thus, although this task is arguably harder with cloud storage than with email, the traffic to the storage service can still be monitored to unveil the covert channel. For instance, the censor may conduct website fingerprinting attacks in order to correlate traffic from a CloudTransport client with traffic patterns that emerge when web-browsing blocked destinations. A way to mitigate this vulnerability is to employ some of the traffic morphing techniques previously discussed in Section 3.3.4. However, as we have stated before, such countermeasures may also be detected by a censor.

Another shortcoming of CloudTransport is that the techniques proposed to help clients to discover and connect to bridges are prone to several attacks. First, the censor may create false bridges to divert traffic to itself and then match the observed traffic with traffic produced by clients to identify the client of a given request. In fact, by performing a Sybil attack [43] an adversary may easily dominate the number of advertised bridges in the system, making this attack very powerful. Also, shall it act like a client, the censor can monitor all accesses to its cloud storage account to obtain the IP addresses of the bridges it interacts with. This enables the censor to neutralize the bridges by staging targeted denial of service attacks.

Further staged protocol tunneling approaches may aim at the mitigation of attacks that leverage the analysis of traffic patterns, by exhibiting patterns common to regular utilization of the cover system.

4.1.3 Castle

Castle [44] leverages common commands issued in Real-Time Strategy (RTS) games to encode and tunnel covert traffic.

Description: The system employs desktop automation software to issue game commands that carry covert data, providing the map used for the ongoing game is custom-made and known apriori by the system. Castle works by selecting structures or immobilized units, respectively setting rally-points or attempting unit displacement to a given map coordinate. The system's authors propose a combinatorial scheme where a move/rally-point command is issued to different subsets of the available units, giving the possibility to encode more data than if a fixed number of units was chosen.

The receiving Castle client must have a way to interpret the covert data in order to retrieve the original data. Usually, RTS games keep a log of the issued commands in order to save or replay a game. This allows the receiving Castle client to fetch and decode Castle's specially crafted commands. As is, a Castle's client may be used to transmit textual data such as emails or other short messages.

Although a censor cannot infer the commands placed by analyzing the encrypted game packets, the traffic flow variance caused by several game factors may be analyzed by the censor. However, this variance is still within the margins of resemblance to an actual human player.

Advantages: Circumvention tools may adjust Castle's design to make use of a panoply of games available to the general public, such as free games like 0 A.D or some of the best-selling RTS games ever. Since many of such games share elements inherent to the RTS genre, a censor gains little by banning a specific game, since another similar one may be used as cover. Thus, the only censor's alternative is to blanket ban all RTS games, incurring in social discontentment.

The games which Castle relies upon typically encrypt and authenticate their network communication channels to prevent cheating, thus preventing the ability of a censor to observe the commands issued by the clients. It also thwarts active attacks relying on rogue packet injection. Moreover, reliable data transmission channels are implemented in the application layer as the majority of RTS games transfer game data through UDP. Therefore, the system tolerates active attacks comprising the drop and delay of packets.

The system's authors propose a different mode of operation where a client can place a request for a web page which would be served by parallel data transfers provided by the remaining clients, acting like a web proxy. This is made possible because several RTS games allow multiple players to join a session.

Limitations: Castle's performance is limited by two factors: i) throughput is highly dependent on the time that the desktop automation tool takes to perform the unit selection and to select a coordinate; ii) different games have different limits on the amount of units that can be selected at once, varying the amount of data that is possible to encode in a given command. Such limitations may prevent a given Castle's implementation to sustain a sufficient throughput for bulk file transfers.

4.1.4 *meek*

meek [45] leverages domain fronting to tunnel traffic over HTTPS connections to allowed hosts, while establishing a covert connection to a prohibited host.

Description: Domain fronting is applied by using different domain names at different layers of the HTTPS request. In such requests, the destination's domain name may be found in three locations: the DNS query, the TLS Server Name Indication (SNI) extension and in the HTTP Host header. While a censor may observe the domain name associated with the DNS query and SNI, it is unable to check the value of the HTTP Host header, since it is hidden by the HTTPS request encryption. This allows for a domain-fronted request to use an allowed domain name on the layers that a censor is able to observe, while hiding the true desired destination in the HTTP Host header. When the frontend server receives such a request, it routes it to the covert destination indicated in the HTTP Host header. Content Distribution Networks (CDNs) are good candidates for deploying frontend servers. As part of their normal operation, they already forward requests to the domain found in HTTP Host header whenever they are unable to serve a request from their local cache.

Advantages: CDNs are widely used today and it may be impossible for a censor to block its usage. Using this strategy, a client may perform a request to a CDN's apparently inoffensive front domain, which will resolve to a frontend server the censor is not able to observe.

Limitations: The patterns observed on *meek*'s TCP ACK traffic are distinct from regular TLS connections. This is due to the fact that *meek*'s clients poll the *meek* server, checking if there is data to be received. It has been shown that it is possible to use machine learning to perform traffic classification in order to distinguish *meek*'s implementation over Tor on particular settings, like a campus network and home wireless networks [38]. Such classifiers work well when deployed in the environments where they were trained, rendering it an interesting approach to be deployed in a localized environment when a censor attempts to confirm a given user is evading censorship.

4.2 Protocol Tunneling Using Multimedia Streaming Applications

The protocol tunneling systems described up to this point leverage a purely staged communication approach, where an oblivious server relays the communication between the client and server of a given censorship circumvention system. A different tunneling approach leveraging multimedia streaming protocols can make use of an oblivious server to relay the communication or take advantage of peer-to-peer communication between the endpoints engaging in circumvention.

To the best of our knowledge, the approach of protocol tunneling through multimedia streaming applications has been explored by two systems, which exhibit remarkable differences between their security properties and use cases. We present both systems on the next paragraphs, highlighting both their advantages and limitations.

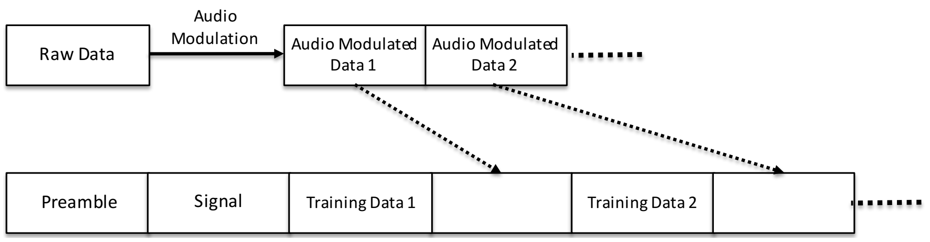

Fig. 3. FreeWave's data frame layout (adapted from FreeWave [9]).

4.2.1 FreeWave

FreeWave [9] leverages VoIP connections to tunnel Internet traffic, allowing for uncensored web browsing.

Description: FreeWave's audio data is generated by a bit-interleaved coded modulation. The system uses a wrapper protocol to carry the modulated data. This wrapper allows for the demodulator to synchronize with the modulator and to negotiate the modulation parameters, necessary to demodulate the acoustic signal. The modulated bit stream is split in data frames. Each data frame has a preamble block which is used for synchronization. The preamble is followed by a signal block which contains the parameters used for the modulation of the data frame. Lastly, the data frame contains interleaved blocks of training data and actual data, as depicted in Figure 3.

FreeWave's modulated data interpretation needs to be synchronized. While the VoIP channel resists to packet dropping, this may prevent the FreeWave's modem from synchronizing. As previously stated, data frames must contain a known preamble block, so that the demodulator can synchronize and interpret the data being sent. Providing that the preamble is not found by the demodulator, it will not be able to decode the actual data. A larger size of the preamble will render a higher probability of a correct identification of a data frame's starting point. One can picture diverse modes of operation, where both the preamble and signal blocks are included: i) uniquely at the beginning of the connection; ii) at the beginning of scheduled data frames with a fixed data blocks number; iii) at each data frame sent whenever the modem has data to send.

Advantages: FreeWave allows the transmission of Internet traffic by modulating it into an audio signal sent through VoIP systems.

The vast number of available VoIP providers makes it hard for a censor to ban all instances of FreeWave, unless it performs a potentially undesirable blanket ban over all VoIP services within its borders.

Limitations: FreeWave can be detected by employing passive attacks and have its functioning thwarted by active attacks, when it is used over VBR codecs. Despite a packet's payload being ciphered, in-depth analysis of the packets' length distribution over time is known to be able to distinguish between the language being spoken in a conversation [46] and even to reconstruct spoken phrases [47]. FreeWave's detection can be achieved based on the observation that the generated network packets' length distribution is nothing similar to that of a recognizable language, expected to be found in an actual conversation [41].

A censor may also prevent FreeWave's modem to synchronize by actively perturbing the network. Indeed, the censor may be able to block any of the three modes of operation described, shall it be able to selectively drop the packets which carry the preamble.

4.2.2 Facet

Facet [10] leverages video conferencing connections to tunnel censored videos.

Description: In order to transmit the content in real time, Facet is built upon a pipeline which downloads and convert the desired video. Instead of using a real microphone and camera, the system makes use of emulators which will be fed with the pipeline data. An interesting insight on Facet is that it avoids channel mismatches by tunneling videos over a video transmission channel. This provides an active attack resistance by design, since any perturbation in the network will cause exactly the same effect on a regular or covert video transmission.

Facet employs *video morphing*, a technique developed to ensure that the network packets generated by the video conferencing software do not directly reflect the characteristics of the censored video, but approximate those of regular video calls instead. To this end, Facet embeds the censored video in a portion of each frame, filling the remaining space with a chat video. This is depicted in Figure 4. A user will then watch a scaled-down censored video over a background chat video. Naturally, the more scaled-down the censored video is, the better the resilience to traffic analysis. This happens because the background video characteristics dominate over those of the embedded censored video.

To foil a censor's traffic classifier results, audio morphing is also required. The audio layer of the censored videos is re-sampled to simulate the lower quality of chat audio.

A Facet's user must communicate with the server in order to specify the URL of the video he wishes to watch. Facet's authors anticipate the blocking of video search services by the censor, giving the possibility for using the Facet server as a proxy for obtaining such URL lists. The user may send an encrypted or steganographically marked email directed at the server (a similar approach to the one discussed in SWEET [21]) to which the server will answer.

Advantages: The described approach yields good enough results to prevent a censor to easily block Facet's sessions. A censor who tries to completely disrupt Facet from functioning incurs into a twenty percent block over legitimate video conferencing calls, when dealing with a scale factor of 0,125. This factor may be adjusted upwards, enabling a user to watch a censored video with better quality, while still representing possible prohibitive collateral damage to the censor.

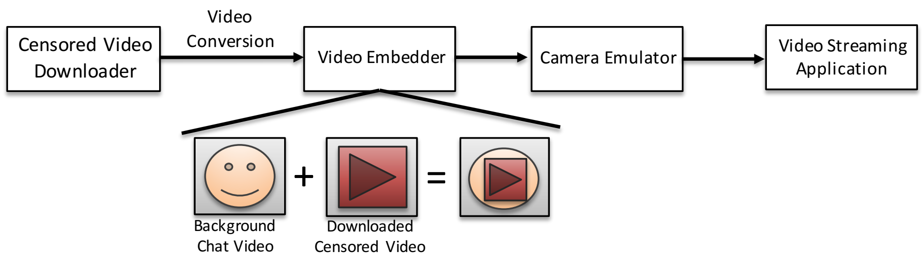
Censored Video Downloader
Video Conversion
Video Embedder
Camera Emulator
Video Streaming Application
Background Chat Video
Downloaded Censored Video

Fig. 4. Facet pipeline overview.

Limitations: By design, the only type of content that Facet is able to serve is video. This limits the system's applicability to other types of communication, such as web browsing. This is because different kinds of data would need to be properly encoded in order to be transmitted, and a reliability layer would need to be in place to resist against active attacks perpetrated by a censor.

Moreover, in order to guarantee a prohibitive collateral damage for the censor, the covert video's scale factor must be small overall, damaging the quality perceived by the client. While video quality may be enough for watching a film's trailer, small scale factors may hinder the viewing experience for several videos. For instance, tutorial-like videos often require the viewer to follow steps shown on-screen. These would be hard, if not impossible, to perceive under high traffic analysis resistance guarantees.

4.3 Discussion

Protocol tunneling systems that create covert channels through staged network applications present good approaches at censorship circumvention. In particular, *meek* is at the forefront of currently deployed protocol tunneling systems, exhibiting a combination of many of the required features to be considered an excellent choice for evading censorship. However, we argue that the encoding of data in multimedia streaming protocols is still relatively unexplored in the context of Internet censorship circumvention. Novel techniques may be devised in order to allow the reliable transmission of arbitrary types of data, while being resilient to active and passive attacks perpetrated by a censor.

FreeWave provides the possibility for arbitrary data transmission over a loss tolerant channel. However, it is vulnerable to active attacks perpetrated by a censor, rendering such transmission unreliable or even impossible at all. Furthermore, FreeWave is unable to resist traffic analysis when used over applications employing VBR codecs, since the network traffic generated by its modulated data is distinguishable from that generated with real speech.

Contrary to FreeWave, Facet is not intended for interactive communication. Although the system's approach seems promising from a traffic analysis resistance point of view, it is limited on the transmission of video content.

Table 1. Protocol tunneling systems comparison.

System	Active Attack Resistance	Passive Attack Resistance	Arbitrary Data Transmission	Interactive Communication	High Throughput
SWEET	Yes	No (limited by utilization mismatch)	Yes	No (limited by utilization mismatch)	Yes
CloudTransport	Yes	No	Yes	Yes	Yes
Castle	Yes	Yes	Yes	Yes	No
meek	Yes	Yes	Yes	Yes	Yes
FreeWave	No	No (w/ VBR codecs)	Yes	Yes	Yes
Facet	Yes	Yes	No	No	Yes

In order to be effective from a security point of view, a censorship circumvention system must exhibit resilience to active and passive attacks from a censor. Moreover, a system that aims to offer both web browsing capability and bulk data transfer must be able to attain certain performance guarantees, namely to achieve a high throughput and to transmit arbitrary sequences of data frames in an interactive way. In Table 1, we depict the surveyed protocol tunneling-based circumvention systems, regarding the aforementioned security and performance properties. A more comprehensive study over the security and performance properties exhibited by existing censorship circumvention systems was presented by Elahi et al. [48].

5 DeltaShaper

In this section we briefly describe the architecture of the censorship circumvention system that we plan to build. The system, named DeltaShaper, encodes data into the video layer of an allowed video conferencing call in order to tunnel covert traffic.

5.1 Data Encoding Approach

DeltaShaper introduces a novel technique which enables the encoding of arbitrary data frames in a video stream, by effectively synthesizing a video frame from the original data.

A first approach, taking inspiration from Facet's video morphing insight, would be to overlay text containing the covert data, expecting that the background video characteristics dominate over those of the overlayed video. Such text could be recovered resorting to Optical Character Recognition (OCR) techniques. However, off-the-shelf OCR solutions can be error prone, even on uniform backgrounds. Suitable image processing and error correction algorithms would need to be devised to recover the overlayed text over a changing background. The study of such algorithms are out of the scope of this work.

A second approach would be to leverage Facet's video morphing technique to transmit video-encoded data, while taking advantage of its traffic analysis resistance design. However, the majority of the channel's bandwidth would be used to transmit the cover background video instead of the data in question.

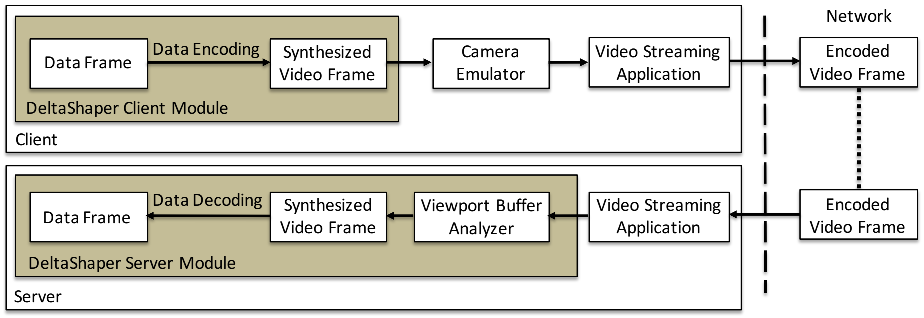

Fig. 5. Video stream synthesis overview.

A more interesting approach would be to use all of the carrier video bandwidth for covert data transfer, by encoding data in each of the synthesized video frames. Careful crafting of such video frames may be able to produce close-to-regular chat video network traffic, while achieving a high bandwidth covert channel. Unlike Facet, this would allow to use an entire video frame to encode covert data, not needing a real chat video to be transmitted as cover.

By synthesizing video frames, the system becomes independent from the video conferencing tool itself. In fact, some of the most popular video conferencing services are proprietary, where the tool's source code is not disclosed and modifications are not allowed. In order to retrieve the original data, DeltaShaper must analyze the video as displayed in each end host's screen. Since video data is commonly ciphered prior to transmission, being deciphered by the application at the receiving end, the video stream receiver only has access to the video signal as displayed by the conferencing application viewport. This mechanism is depicted in Figure 5.

We focus our work on developing the proposed video synthesis technique, which surges as a base to deploy a covert channel over a video conferencing call between two parties engaging in circumvention. The system is designed to be interactive, allowing a client to send requests and to receive the corresponding responses, laying the foundations for web-browsing capability. Although peer discovery and the tool's distribution must be considered when deploying a censorship circumvention system, these are issues outside of the scope of this work.

5.2 Supporting Application Data Transmission

The DeltaShaper server may run a SOCKS [49] proxy in order to direct the network traffic of its clients to their desired destinations. The SOCKS protocol enables DeltaShaper to tunnel several application protocols, such as HTTP to support web-browsing or FTP to support bulk file transfers.

A user of DeltaShaper may configure an application (e.g. web browser or FTP client) to send its traffic through the SOCKS protocol. Instead of directly contacting the SOCKS server proxy, the user application will direct its SOCKS

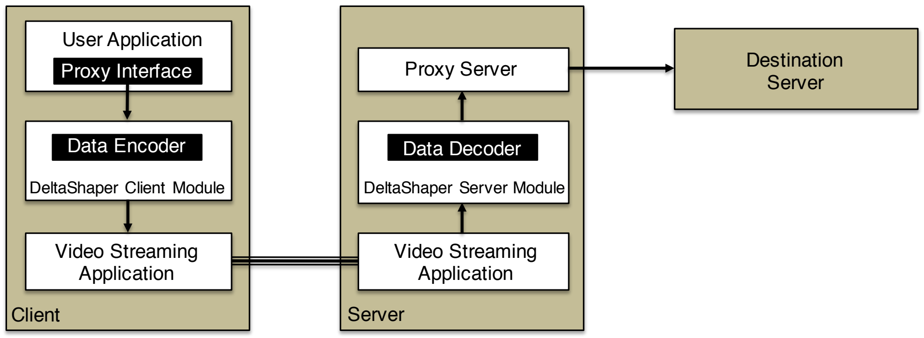
User Application
Proxy Interface
Data Encoder
DeltaShaper Client Module
Video Streaming Application
Client
Proxy Server
Data Decoder
DeltaShaper Server Module
Video Streaming Application
Server
Destination Server

Fig. 6. Application data transmission overview.

traffic to DeltaShaper's local client, which will encode the protocol messages into video frames and send them through the chosen video conferencing system. The DeltaShaper's server shall decode and deliver the received messages to a SOCKS proxy server, running on the same machine, which will then communicate with the client's desired destination. The response shall be delivered to the client in a similar way. This mechanism is depicted on Figure 6.

In order to support the tunneling of SOCKS data, a wrapper protocol must be implemented so as to assure the reliability of the transmission. This protocol may also be responsible for multiplexing connection data from one to several applications at once, between the DeltaShaper client and server.

5.3 On Avoiding Covert and Carrier Channels Mismatches

As discussed in 3.3.6, protocol tunneling systems face four types of mismatches between the covert and cover channel. The following paragraphs describe the approach followed by DeltaShaper in order to resist to attacks that leverage such mismatches:

Architectural Mismatch Prevention: DeltaShaper exhibits the same limitations observed on previous similar systems described in the literature, regarding architectural mismatches. Concretely, the use of a decentralized video conferencing system may turn a DeltaShaper server into a hot spot, since a censor will be able to observe several video conferencing connections to it. If DeltaShaper users choose a centralized video conferencing system instead, their calls are relayed through an oblivious server which may also be relaying regular calls. This may prevent a censor from blocking such oblivious relay servers (and consequently DeltaShaper) due to potential collateral damage.

Content Mismatch Prevention: In widespread video streaming protocols such as VP8 or H.264, updates between subsequent video frames are mostly performed through commonly called delta frames [50, 51], which provide data about the incremental changes between adjacent frames.

In order to provide “unobservability”, DeltaShaper’s synthesized video frames should keep the difference to their directly related frames below a certain threshold. To prevent a content mismatch, the rate and size of the generated delta frames must not be distinguishable from that of a regular chat video by an observer of the network traffic.

The audio layer also poses as a concern from a traffic analysis perspective. Therefore, users engaging in circumvention can maintain an actual conversation or recorded audio may be played so that the generated network traffic reflects that of a regular VoIP conversation.

Channel Mismatch Prevention: The transfer of a data stream through a loss tolerant channel such as video calls raises the issue of channel mismatches, from which a censor may take advantage. In order to tolerate active attacks aimed at corrupting the covert data transfer, DeltaShaper’s wrapper protocol shall also be responsible for providing reliability on the application layer.

Utilization Mismatch Prevention: We argue that a video call provides cover during enough time for an average web-browsing session to take place, without raising suspicion from a censor. However, we emphasize that this is the only mismatch that directly depends on a responsible and informed utilization of the censorship circumvention system by its users.

6 Evaluation

We intend to evaluate our prototype regarding two main aspects: i) performance; ii) resilience against passive and active attacks.

6.1 Performance Evaluation

The performance of our system will be evaluated by measuring the achieved throughput and latency, taking into account different kinds of traffic, such as the one generated by bulk file transmissions and web-browsing traffic.

We expect the throughput of our system to surpass the one provided by FreeWave, since the bandwidth available for a video stream channel is larger than that available for an audio channel. We also expect the latency of our system to be sufficient for supporting interactive communication like web-browsing.

Naturally, both performance measures are likely to be affected by the amount of data that can be encoded on a video frame, providing that the rate and size of video frames’ updates observed on the network are similar to those of a regular chat video.

6.2 Attack Resilience Evaluation

We shall evaluate the system’s resilience against both active and proactive attacks. As in the system’s performance evaluation, we shall measure attack resilience when tunneling different kinds of data.

To resist a passive attack, the network traces generated by the video conferencing service should not be dramatically different when establishing a regular video conferencing call or a call where our system is being employed. As discussed previously in the text, this is a prevalent issue when dealing with VBR codecs. To this end, we shall collect network traces of a multitude of both types of call in order to employ statistical traffic analysis techniques. In particular, a χ^2 binary classifier [10, 46] shall be employed to determine whether a DeltaShaper video call is considered legitimate or not. We shall measure the false positive and false negative rates of the classifier so as to understand how many legitimate calls would be shut down by the censor, shall it aim to disrupt our system from functioning. We expect the traffic analysis resistance offered by our system to at least match the one provided by Facet, since we expect to have a finer-grained control over the video signal itself.

In order to evaluate the resistance of our system against an active attack perpetrated by the censor, we shall measure the time it takes for our system to resume a given covert data transfer after active perturbation on the network, as well as the correctness of the transferred data itself.

7 Scheduling of Future Work

Future work is scheduled as follows:

- January 9 - March 29: Detailed design and implementation of the proposed architecture, including preliminary tests.
- March 30 - May 3: Perform the complete experimental evaluation.
- May 4 - May 23: Write a paper describing the project.
- May 24 - June 15: Finish the writing of the dissertation.
- June 15: Deliver the MSc dissertation.

8 Conclusions

Repressive regimes actively censor the access to the Internet, foiling the ability for their citizens to obtain or to publish rightful information. In this work we have surveyed the relevant related work regarding both censorship enforcing and circumvention techniques.

We propose a novel data encoding technique through video stream synthesis which can be leveraged to build a covert channel over popular video conferencing applications, overcoming limitations exhibited by previous systems.

We have presented the architecture of the proposed solution. Its detailed specification, implementation, and experimental evaluation are left for future work, whose schedule has also been presented.

References

1. Aryan, S., Aryan, H., Halderman, J.A.: Internet censorship in Iran : A first look. In: Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet, Washington, DC, USA (2013)
2. Chaabane, A., Chen, T., Cunche, M., De Cristofaro, E., Friedman, A., Kaafar, M.A.: Censorship in the wild: Analyzing Internet filtering in Syria. In: Proceedings of the 2014 Conference on Internet Measurement Conference, Vancouver, BC, Canada (2014) 285–298
3. Knockel, J., Crete-Nishihata, M., Ng, J.Q., Senft, A., Crandall, J.R.: Every rose has its thorn: Censorship and surveillance on social video platforms in China. In: Proceedings of the 5th USENIX Workshop on Free and Open Communications on the Internet, Washington, D.C., USA (2015)
4. Dainotti, A., Squarcella, C., Aben, E., Claffy, K.C., Chiesa, M., Russo, M., Pescapé, A.: Analysis of country-wide Internet outages caused by censorship. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, Berlin, Germany (2011)
5. Syria Goes Offline. `http://new.secdev-foundation.org/wp-content/uploads/2014/08/Flash-Note-Syria-2-Syria-goes-offline.pdf` Accessed: 2015-11-19.
6. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium. (2004)
7. A Child's Garden Of Pluggable Transports. `https://trac.torproject.org/projects/tor/wiki/doc/AChildsGardenOfPluggableTransports\#OrdinaryTor` Accessed: 2015-11-19.
8. Houmansadr, A., Brubaker, C., Shmatikov, V.: The parrot is dead: Observing unobservable network communications. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy, San Francisco, CA, USA (2013) 65–79
9. Houmansadr, A., Riedl, T.J., Borisov, N., Singer, A.C.: I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention. In: Proceedings of the 20th Annual Network & Distributed System Security Symposium. (2013)
10. Li, S., Schliep, M., Hopper, N.: Facet: Streaming over videoconferencing for censorship circumvention. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society, Scottsdale, Arizona, USA (2014) 163–172
11. OpenNet Initiative - China's Green Dam: The Implications of Government Control Encroaching on the Home PC. `https://opennet.net/sites/opennet.net/files/GreenDam_bulletin.pdf` Accessed: 2015-12-05.
12. Wagner, B.: Deep packet inspection and Internet censorship: International convergence on an 'integrated technology of control'. In: Proceedings of the 3rd Annual Giganet Symposium, Hyderabad, India (2008)
13. Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., Karger, D.: Infranet: Circumventing web censorship and surveillance. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA (2002) 247–262
14. Burnett, S., Feamster, N., Vempala, S.: Chipping away at censorship firewalls with user-generated content. In: Proceedings of the 19th USENIX Conference on Security, Washington, DC, USA (2010)
15. Wustrow, E., Wolchok, S., Goldberg, I., Halderman, J.A.: Telex: Anticensorship in the network infrastructure. In: Proceedings of the 20th USENIX Conference on Security, San Francisco, CA, USA (2011)

16. Houmansadr, A., Nguyen, G.T., Caesar, M., Borisov, N.: Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, Chicago, Illinois, USA (2011) 187–200
17. Moghaddam, H., Li, B., Derakhshani, M., Goldberg, I.: Skypemorph: Protocol obfuscation for Tor bridges. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA (2012) 97–108
18. Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Cheung, S., Wang, F., Boneh, D.: Stegotorus: A camouflage proxy for the Tor anonymity system. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA (2012) 109–120
19. Wang, Q., Gong, X., Nguyen, G.T., Houmansadr, A., Borisov, N.: Censorspoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA (2012) 121–132
20. Brubaker, C., Houmansadr, A., Shmatikov, V.: Cloudtransport: Using cloud storage for censorship-resistant networking. In De Cristofaro, E., Murdoch, S., eds.: Privacy Enhancing Technologies. Volume 8555 of Lecture Notes in Computer Science. Springer International Publishing (2014) 1–20
21. Zhou, W., Houmansadr, A., Caesar, M., Borisov, N.: Sweet: Serving the web by exploiting email tunnels. In: Proceedings of the 6th Workshop on Hot Topics in Privacy Enhancing Technologies. (2013)
22. Tanash, R.S., Chen, Z., Thakur, T., Wallach, D.S., Subramanian, D.: Known unknowns: An analysis of twitter censorship in Turkey. In: Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, Denver, Colorado, USA (2015) 11–20
23. Cheddad, A., Condell, J., Curran, K., Kevitt, P.M.: Digital image steganography: Survey and analysis of current methods. In: Signal Processing 90(3). (2010) 727–752
24. Djebbar, F., Ayad, B., Meraim, K.A., Hamam, H.: Comparative study of digital audio steganography techniques. In: EURASIP Journal on Audio, Speech, and Music Processing. (2012) 1–16
25. Li, B., He, J., Huang, J., Shi, Y.Q.: A survey on image steganography and steganalysis. In: Journal of Information Hiding and Multimedia Signal Processing, 2(2). (2011) 142–172
26. Ultrasurf. `http://ultrasurf.us` Accessed: 2015-11-19.
27. Karlin, J., Ellard, D., Jackson, A., Jones, C., Lauer, G., Mankins, D., Strayer, T.: Decoy routing: Toward unblockable Internet communication. In: Proceedings of the USENIX Workshop on Free and Open Communications on the Internet, San Francisco, CA, USA (2011)
28. Schuchard, M., Geddes, J., Thompson, C., Hopper, N.: Routing around decoys. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA (2012) 85–96
29. Houmansadr, A., Wong, E.L., Inc, M., Shmatikov, V.: No direction home: The true cost of routing around decoys. In: Proceedings of the 21th Annual Network & Distributed System Security Symposium, San Diego, CA, USA (2014)
30. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, Chicago, Illinois, USA (2009) 31–42

31. Wright, C.V., Coull, S.E., Monrose, F.: Traffic morphing: An efficient defense against statistical traffic analysis. In: Proceedings of the 16th Network and Distributed Security Symposium, San Diego, CA, USA (2009)
32. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, Chicago, Illinois, USA (2011) 103–114
33. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: Website fingerprinting attacks and defenses. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, North Carolina, USA (2012) 605–616
34. Obfsproxy. `https://www.torproject.org/projects/obfsproxy.html.en` Accessed: 2015-11-19.
35. Winter, P., Pulls, T., Fuss, J.: Scramblesuit: A polymorphic network protocol to circumvent censorship. In: Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society, Berlin, Germany (2013) 213–224
36. Pluggable Transports. `https://gitweb.torproject.org/torspec.git/tree/pt-spec.txt` Accessed: 2015-11-19.
37. RFC 6246 - TLSHandshake. `https://tools.ietf.org/html/rfc5246` Accessed: 2015-11-19.
38. Wang, L., Dyer, K.P., Akella, A., Ristenpart, T., Shrimpton, T.: Seeing through network-protocol obfuscation. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, Colorado, UK (2015) 57–69
39. Dyer, K.P., Coull, S.E., Ristenpart, T., Shrimpton, T.: Protocol misidentification made easy with format-transforming encryption. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany (2013) 61–72
40. Dyer, K.P., Coull, S.E., Shrimpton, T.: Marionette: A programmable network-traffic obfuscation system. In: Proceedings of the 24th USENIX Conference on Security Symposium, Washington, D.C., USA (2015) 367–382
41. Geddes, J., Schuchard, M., Hopper, N.: Cover your acks: Pitfalls of covert channel censorship circumvention. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin, Germany (2013) 361–372
42. Skype Company Statistics - 2015 Statistic Brain Research Institute, publishing as Statistic Brain. `http://www.statisticbrain.com/skype-statistics/` Research Date: June 6th, 2015, Accessed: 2015-11-19.
43. Douceur, J.R.: The Sybil attack. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems, London, UK (2002) 251–260
44. Hahn, B., Nithyanand, R., Gill, P., Johnson, R.: Games without frontiers: Investigating video games as a covert channel. In: arXiv:1503.05904 [cs.CR]. (2015)
45. Fifield, D., Lan, C., Hynes, R., Wegmann, P., Paxson, V.: Blocking-resistant communication through domain fronting. In: Proceedings on Privacy Enhancing Technologies 2015.2, Philadelphia, PA, USA (2015) 46–64
46. Wright, C.V., Ballard, L., Monrose, F., Masson, G.M.: Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In: Proceedings of 16th USENIX Security Symposium, Boston, MA (2007) 4:1–4:12
47. White, A.M., Matthews, A.R., Snow, K.Z., Monrose, F.: Phonotactic reconstruction of encrypted VoIP conversations: Hookt on fon-iks. In: Proceedings of the 2011 IEEE Symposium on Security and Privacy, Oakland, CA, USA (2011) 3–18

48. Elahi, T., M. Swanson, C., Goldberg, I.: Slipping past the cordon: A systematization of Internet censorship resistance. In: CACR Tech Report 2015-10. (2015)
49. RFC 1928 - SOCKS Protocol Version 5. `https://tools.ietf.org/html/rfc1928` Accessed: 2015-12-14.
50. RFC 6386 - VP8 Data Format and Decoding Guide. `https://tools.ietf.org/html/rfc6386` Accessed: 2015-12-05.
51. Juurlink, B., Alvarez-Mesa, M., Chi, C.C., Azevedo, A., Meenderinck, C., Ramirez, A.: Scalable parallel programming applied to H.264/AVC decoding. In: SpringerBriefs in Computer Science. (2012)