Internet Engineering Task Force (IETF)
Request for Comments: 5780
Category: Experimental
ISSN: 2070-1721

D. MacDonald
B. Lowekamp
Skype

May 2010

NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)

Abstract

This specification defines an experimental usage of the Session Traversal Utilities for NAT (STUN) Protocol that discovers the presence and current behavior of NATs and firewalls between the STUN client and the STUN server.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5780.

Table of Contents

1. Applicability

This experimental NAT Behavior Discovery STUN usage provides information about a NAT device's observable transient behavior; it determines a NAT's behavior with regard to the STUN server used and the particular client ports used at the instant the test is run. This STUN usage does not allow an application behind a NAT to make an absolute determination of the NAT's characteristics. NAT devices do not behave consistently enough to predict future behavior with any guarantee. Applications requiring reliable reach between two particular endpoints must establish a communication channel through NAT using another technique. IETF has proposed standards including [RFC5245] and [RFC5626] for establishing communication channels when a publicly accessible rendezvous service is available.

The uses envisioned for the STUN attributes included in this document are diagnostics and real-time tuning of applications. For example, determining what may work and should be tried first compared to more expensive methods. The attributes can also be used to observe behaviors that causes an application's communication to fail, thus enabling better selection of methods of recovery. The STUN attributes could also be a basis for a network technician's diagnostics tool to observe NAT behavior.

This document proposes experimental usage of these attributes for real-time optimization of parameters for protocols in situations where a publicly accessible rendezvous service is not available. Such a use of these techniques is only possible when the results are applied as an optimization and a reliable fallback is available in case the NAT's behavior becomes more restrictive than determined by the Behavior Discovery tests. One possible application is role selection in peer-to-peer (P2P) networks based on statistical experience with establishing direct connections and diagnosing NAT behavior with a variety of peers. The experimental question is whether such a test is useful. Consider a node that tries to join an overlay as a full peer when its NAT prevents sufficient connectivity; joining and withdrawing from the overlay might be expensive and/or lead to unreliable or poorly performing operations. Even if the behavior discovery check is only "correct" 75% of the time, its relative cheapness may make it very useful for optimizing the behavior of the overlay network. Section 2.2 describes this experimental application in more detail and discusses how to evaluate its success or failure.

The applications of this STUN usage differ from the original use of STUN (originally RFC 3489 [RFC3489], now RFC 5389 [RFC5389]). This specification acknowledges that the information gathered in this

usage is not, and cannot be, correct 100% of the time, whereas STUN focused only on getting information that could be known to be correct and static.

This specification can also be compared to ICE. ICE requires a fallback to TURN be available whereas RFC 3489 based applications tried to determine in advance whether they would need a relay and what their peer reflexive address will be, which is not generally achievable.

This STUN usage requires an application using it to have a fallback. However, unlike ICE's focus on the problems inherent in VoIP sessions, this STUN usage doesn't assume that it will be used to establish a connection between a single pair of machines, so alternative fallback mechanisms may be available.

For example, in a P2P application it may be possible to simply switch out of the role where such connections need to be established or to select an alternative indirect route if the peer discovers that, in practice, 10% of its connection attempts fail.

It is submitted to the Internet community as an experimental protocol that, when applied with appropriate statistical underpinnings and application behavior that is ultimately based on experienced connectivity patterns, can lead to more stability and increased performance than is available without the knowledge it provides.

If a Standards Track document specifies the use of any portion of this STUN usage, that document MUST describe how incorrect information derived using these methods will be managed, either through identifying when a NAT's behavior changed or because the protocol uses such knowledge as an optimization but remains functional when the NAT's behavior changes. The referencing document MUST also define when the fallback mechanism will be invoked. Applications in different domains may vary greatly in how aggressively the fallback mechanism is utilized, so there must be a clear definition of when the fallback mechanism is invoked.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Introduction

"Session Traversal Utilities for NAT (STUN)" [RFC5389] provides a mechanism to discover the reflexive transport address toward the STUN server, using the Binding Request. This specification defines the NAT Behavior Discovery STUN usage, which allows a STUN client to probe the current behavior of the NAT/firewall (NAT/FW) devices between the client and the STUN server. This usage defines new STUN attributes for the Binding Request and Binding Response.

Many NAT/FW devices do not behave consistently and will change their behavior under load and over time. Applications requiring high reliability must be prepared for the NAT's behavior to become more restrictive. Specifically, it has been found that under load NATs may transition to the most restrictive filtering and mapping behavior and shorten the lifetime of new and existing bindings. In short, applications can discover how bad things currently are, but not how bad things will get.

Despite this limitation, instantaneous observations are often quite useful in troubleshooting network problems, and repeated tests over time, or in known load situations, may be used to characterize a NAT's behavior. In particular, in the hands of a person knowledgeable about the needs of an application and the nodes an application needs to communicate with, it can be a powerful tool.

2.1. Example Diagnostic Use

Applications that work well in the lab, but fail in a deployment, are notoriously common within distributed systems. There are few systems developers who have not had the experience of searching to determine the difference in the environments for insight as to what real-network behavior was missed in the testing lab. The Behavior Discovery usage offers a powerful tool that can be used to check NAT and firewall behavior as the application is running. For example, an application could be designed to perform Behavior Discovery tests whenever it experiences significant communications problems when running. Such analysis might be included as part of the diagnostic information logged by the application.

As they are being used to detect instantaneous behavior for analysis by an experienced developer or administrator, there are relatively few concerns about this application of the NAT Behavior Discovery STUN usage. However, the user should be aware that

- adding new traffic to new destinations (STUN servers) has the potential to itself change the behavior of a NAT and

- o the user must be careful to select a STUN server that is appropriately located, ideally collocated (or even integrated) with the communication partners of the application in question, for the results to be applicable to the network conditions experienced by the application.

2.2. Example Use with P2P Overlays

An application could use Behavior Discovery in a P2P protocol to determine if a particular endpoint is a reasonable candidate to participate as a peer or supernode (defined here as a peer in the overlay that offers services, including message routing, to other members or clients of the overlay network). This P2P network application is willing to select supernodes that might be located behind NATs to avoid the cost of dedicated servers. A supernode candidate requires that its NAT or NATs offer Endpoint-Independent Filtering. It might periodically re-run tests and would remove itself as a supernode if its NAT/FW chain lost this characteristic. These tests could be run with other supernodes acting as STUN servers as well as with dedicated STUN servers. As many P2P algorithms tolerate non-transitive connectivity between a portion of their peers, guaranteed pair-wise reliable reach might be sacrificed in order to distribute the P2P overlay's load across peers that can be directly contacted by the majority of users.

Consider an example from a hypothetical P2P protocol in more detail: when P2P node A starts up, it tests its NAT(s) relative to other peers already in the overlay. If the results of its testing indicate A is behind a "good" NAT (with Endpoint-Independent Mapping and Filtering), A will join the overlay and establish connections with appropriate peers in the overlay to join the overlay's topology. Although A is reachable by routing messages across the overlay topology, A will also include in its communication with other nodes that they may reach it directly using its reflexive IP address (or addresses) that A discovered in its initial testing. Suppose that later, node B wants to send a message to A, and B is not a neighbor of A in the overlay topology. B may send the message directly to A's IP address and start a timer. If B doesn't receive a response within a certain amount of time, then it routes the message to A across the overlay instead and includes a flag that indicates a direct connection was attempted but failed. (Alternatively, B could simultaneously send the message to A's IP address across the overlay, which guarantees minimum response latency, but can waste bandwidth.) Over time, A observes the percentage of successful direct messages it receives out of those attempted. If the percentage of successful direct connections is below some threshold (perhaps 75%), then A may stop advertising for direct connections because it has determined in practice that its NATs are not providing sufficiently reliable

connectivity to justify the cost of attempting the direct message. But if the percentage is high enough, A continues to advertise because the successful direct connections are improving the overlay's performance by reducing the routing load imposed on the overlay. If at some point, A's NAT or NATs change behavior, A will notice a change in its percentage of successful direct connections and may re-evaluate its decision to advertise a public address. In this hypothetical example, behavior discovery is used for A's initial operating mode selection, but the actual decision for whether to continue advertising that public IP/port pair is made based on actual operating data. The results of the Behavior Discovery usage are also used as a performance optimization, as A is at all times able to establish connectivity through the overlay if the attempted direct connection fails.

Use of behavior discovery for such an application requires:

- Use of a protocol capable of offering reliable end-user performance while using unreliable links between pairs of nodes.

- A protocol offering a reliable fallback to connections attempted based on the results of Behavior Discovery probing.

- The application is deployed behind NATs that provide Endpoint-Independent Filtering and that remain in this mode for an amount of time sufficient for the application to identify their behavior, distribute this information to the rest of the overlay, and provide useful work for the application.

This document is experimental as applications implementing open protocols have yet to be deployed in such environments to demonstrate that these three requirements have been met. However, anecdotal evidence suggests that NATs targeted at households and small businesses have stable behavior, especially when there are few clients behind them. Numerous P2P applications have been deployed that appear to have these properties, although their protocols have not yet been subjected to rigorous evaluation by standards bodies.

2.3. Experimental Goals

The criteria for an application to successfully demonstrate use of the NAT Behavior Discovery STUN usage would include:

- An implementation that relies on this usage to determine its run-time behavior, most likely using it to determine an initial choice of options that are then adjusted based on experience with its network connections.

- The implementation must either demonstrate its applicability in environments where it is realistic to expect a provider to deploy dedicated STUN servers with multiple IP addresses, or it must demonstrate duplicating the behavior of such a dedicated STUN server with two nodes that share the role of providing the address-changing operations required by this usage.

- Experimental evidence that the application of this usage results in improved behavior of the application in real-world conditions. The exact metrics for this improvement may vary, some possibilities include: faster convergence to the proper parameters, less work to set up initial connections, fewer reconfigurations required after startup, etc.

- A protocol specification that defines how the implementation applies this usage.

The P2P scenario described above is a likely experimental test case for this usage, but others applications are possible as well.

3. Overview of Operations

In a typical configuration, a STUN client is connected to a private network and through one or more NATs to the public Internet. The client is configured with the address of a STUN server on the public Internet. The Behavior Discovery usage makes use of SRV records so that a server may use a different transport address for this usage than for other usages. This usage does not provide backward compatibility with RFC 3489 [RFC3489] for either clients or servers. Implementors of clients that wish to be compliant with RFC 3489 servers should see that specification. Implementors of servers SHOULD NOT include support for RFC 3489 clients, as the original uses of that protocol have been deprecated.

Because STUN forbids a server from creating a new TCP or TCP/TLS connection to the client, many tests apply only to UDP. The applicability of the various tests is indicated below.

The STUN NAT Behavior Discovery usage defines new attributes on the STUN Binding Request and STUN Binding Response that allow these messages to be used to diagnose the current behavior of the NAT(s) between the client and server.

This section provides a descriptive overview of the typical use of these attributes. Normative behavior is described in Sections 5, 6, and 7.

3.1. Determining NAT Mapping

A client behind a NAT wishes to determine if that NAT is currently using Endpoint-Independent, Address-Dependent, or Address and Port-Dependent Mapping [RFC4787]. The client performs a series of tests that make use of the OTHER-ADDRESS attribute; these tests are described in detail in Section 4. These tests send binding requests to the alternate address and port of the STUN server to determine mapping behavior. These tests can be used for UDP, TCP, or TCP/TLS connections.

3.2. Determining NAT Filtering

A client behind a NAT wishes to determine if that NAT is currently using Endpoint-Independent, Address-Dependent, or Address and Port-Dependent Filtering [RFC4787]. The client performs a series of tests that make use of the OTHER-ADDRESS and CHANGE-REQUEST attributes; these tests are described in Section 4. These tests request responses from the alternate address and port of the STUN server; a precondition to these tests is that no binding be established to the alternate address and port. See below for more information. Because the NAT does not know that the alternate address and port belong to the same server as the primary address and port, it treats these responses the same as it would those from any other host on the Internet. Therefore, the success of the binding responses sent from the alternate address and port indicate whether the NAT is currently performing Endpoint-Independent Filtering, Address-Dependent Filtering, or Address and Port-Dependent Filtering. This test applies only to UDP datagrams.

3.3. Binding Lifetime Discovery

Many systems, such as VoIP, rely on being able to keep a connection open between a client and server or between peers of a P2P system. Because NAT bindings expire over time, keepalive messages must be sent across the connection to preserve it. Because keepalives impose some overhead on the network and servers, reducing the frequency of keepalives can be useful.

A normal request-response protocol cannot be used to test binding lifetime because the initial request resets the binding timer. Behavior discovery defines the RESPONSE-PORT attribute to allow the client and server to set up a "control channel" using one port on the client that is used to test the binding lifetime of a different port allocated on the client. More generally, RESPONSE-PORT allows the client to allocate two ports and request that responses to queries sent from one port be delivered to the other. The client uses its second port and the STUN server’s alternate address to check if an

existing binding that hasn't had traffic sent on it is still open after time T. This approach is described in detail in Section 4.6. This test applies only to UDP datagrams.

3.4. Diagnosing NAT Hairpinning

STUN Binding Requests allow a client to determine whether it is behind a NAT that supports hairpinning of connections. To perform this test, the client first sends a Binding Request to its STUN server to determine its mapped address. The client then sends a STUN Binding Request to this mapped address from a different port. If the client receives its own request, the NAT hairpins connections. This test applies to UDP, TCP, or TCP/TLS connections.

3.5. Determining Fragment Handling

Some NATs exhibit different behavior when forwarding fragments than when forwarding a single-frame datagram. In particular, some NATs do not hairpin fragments at all and some platforms discard fragments under load. To diagnose this behavior, STUN messages may be sent with the PADDING attribute, which simply inserts additional space into the message. By forcing the STUN message to be divided into multiple fragments, the NAT's behavior can be observed.

All of the previous tests can be performed with PADDING if a NAT's fragment behavior is important for an application, or only those tests that are most interesting to the application can be retested. PADDING only applies to UDP datagrams. PADDING can not be used with RESPONSE-PORT.

3.6. Detecting a Generic Application Level Gateway (ALG)

A number of NAT boxes are now being deployed into the market that try to provide "generic" ALG functionality. These generic ALGs hunt for IP addresses, either in text or binary form within a packet, and rewrite them if they match a binding. This behavior can be detected because the STUN server returns both the MAPPED-ADDRESS and XOR-MAPPED-ADDRESS in the same response. If the result in the two does not match, there is a NAT with a generic ALG in the path. This test apples to UDP and TCP, but not TLS over TCP connections.

4. Discovery Process

This section provides a descriptive overview of how the NAT Behavior Discovery usage primitives allow checks to be made to discover the current behavior of the NAT or NATs an application is behind. These tests can only give the instantaneous behavior of a NAT; it has been found that NATs can change behavior under load and over time. The

results of these tests therefore can be regarded as upper bounds -- an application must assume that NAT behavior can become more restrictive at any time. Results from tests performed using a particular port on the client may also not indicate the behavior experienced by a different port, as described in Section 4.1.

Definitions for NAT filtering and mapping behavior are from [RFC4787]. The tests described here are for UDP connectivity, NAT mapping behavior, NAT filtering behavior, and NAT binding lifetime discovery; additional tests could be designed using this usage's mechanisms. The tests described below include only tests that can be performed using a client with a single IP address. A client with multiple IP addresses (or multiple clients collaborating) behind the same NAT can combine their probes to test additional aspects of NAT behavior, such as port overloading. This section provides a descriptive overview of how the primitives provided by the STUN attributes in this specification may be used to perform behavior tests.

Normative specifications for the attributes are defined in later sections.

4.1. Source Port Selection

Proper source port selection is important to ensuring the usefulness and accuracy of the Behavior Discovery tests. There are two preconditions for tests:

- Because mapping behavior can vary on a port-by-port basis, an application should perform its tests using the source port intended for use by the application whenever possible. If it intends to use multiple source ports, it should repeat these tests for each source port. Such tests should be performed sequentially to reduce load on the NAT.

- Because the results of some diagnostic checks depend on previous state in the NAT created by prior traffic, the tests should be performed using a source port that has not generated recent traffic. Therefore, the application should use a random source port or ensure that no traffic has previously occurred on the selected port prior to performing tests, generally by allocating a port and holding it unused for at least 15 minutes prior to the tests.

Ensuring both of these preconditions can be challenging, particularly for a device or application wishing to perform Behavior Discovery tests at startup. The following guidelines are suggested for reducing the likelihood of problems:

- An application intended to operate behind a NAT should not attempt to allocate a specific or well-known port. Because such software must be designed to interoperate using whatever port is mapped to it by the NAT, the specific port is unnecessary. Instead, on startup, a random port should be selected (see below for recommended ranges). An application, particularly on an embedded device, should not rely on the host operating system to select the next available port because that might result in the application receiving the same port on each restart. An application using the same port between restarts may not receive accurate results from Behavior Discovery tests that are intended to test state-related behavior of NATs, such as filtering and binding lifetime.

- An application requiring multiple ports, such as separate ports for control and media, should allocate those ports on startup when possible. Even if there is no immediate need for media flow, if Behavior Discovery tests will be run on those ports, allocating them early will allow them to be left idle, increasing the chance of obtaining accurate results from Behavior Discovery tests.

- Although the most reliable results are obtained when performing tests with the specific ports that the application will use, in many cases an application will need to allocate and use ports without being able to perform complete Behavior Discovery tests on those ports. In those cases, an application should randomly select its ports from a range likely to receive the same treatment by the NAT. This document recommends ranges of 32768-49151, which is the upper end of IANA's Registered Ports range, and 49152-65535, which is IANA's Dynamic and/or Private port range, for random selection. To attempt to characterize a NAT's general treatment of ports in these ranges, a small number of ports within a range can be randomly selected and characterized.

Those tests particularly sensitive to prior state on a NAT will be indicated below.

4.2. Checking for UDP Connectivity with the STUN Server

The client sends a STUN Binding Request to a server. This causes the server to send the response back to the address and port that the request came from. If this test yields no response, the client knows right away that it does not have UDP connectivity with the STUN server. This test requires only STUN [RFC5389] functionality.

4.3. Determining NAT Mapping Behavior

This will require at most three tests. In test I, the client performs the UDP connectivity test. The server will return its alternate address and port in OTHER-ADDRESS in the binding response. If OTHER-ADDRESS is not returned, the server does not support this usage and this test cannot be run. The client examines the XOR-MAPPED-ADDRESS attribute. If this address and port are the same as the local IP address and port of the socket used to send the request, the client knows that it is not NATed and the effective mapping will be Endpoint-Independent.

In test II, the client sends a Binding Request to the alternate address, but primary port. If the XOR-MAPPED-ADDRESS in the Binding Response is the same as test I the NAT currently has Endpoint-Independent Mapping. If not, test III is performed: the client sends a Binding Request to the alternate address and port. If the XOR-MAPPED-ADDRESS matches test II, the NAT currently has Address-Dependent Mapping; if it doesn't match it currently has Address and Port-Dependent Mapping.

4.4. Determining NAT Filtering Behavior

This will also require at most three tests. These tests are sensitive to prior state on the NAT.

In test I, the client performs the UDP connectivity test. The server will return its alternate address and port in OTHER-ADDRESS in the binding response. If OTHER-ADDRESS is not returned, the server does not support this usage and this test cannot be run.

In test II, the client sends a binding request to the primary address of the server with the CHANGE-REQUEST attribute set to change-port and change-IP. This will cause the server to send its response from its alternate IP address and alternate port. If the client receives a response, the current behavior of the NAT is Endpoint-Independent Filtering.

If no response is received, test III must be performed to distinguish between Address-Dependent Filtering and Address and Port-Dependent Filtering. In test III, the client sends a binding request to the original server address with CHANGE-REQUEST set to change-port. If the client receives a response, the current behavior is Address-Dependent Filtering; if no response is received, the current behavior is Address and Port-Dependent Filtering.

4.5. Combining and Ordering Tests

Clients may wish to combine and parallelize these tests to reduce the number of packets sent and speed the discovery process. For example, test I of the filtering and mapping tests also checks if UDP is blocked. Furthermore, an application or user may not need as much detail as these sample tests provide. For example, establishing connectivity between nodes becomes significantly more difficult if a NAT has any behavior other than Endpoint-Independent Mapping, which requires only test I and II of Section 4.3. An application that determines its NAT does not always provide Endpoint-Independent Mapping might notify the user if no relay is configured, whereas an application behind a NAT that provides Endpoint-Independent Mapping might not notify the user until a subsequent connection actually fails or might provide a less urgent notification that no relay is configured. Such a test does not alleviate the need for [RFC5245], but it does provide some information regarding whether ICE is likely to be successful establishing non-relayed connections.

Care must be taken when combining and parallelizing tests, due to the sensitivity of certain tests to prior state on the NAT and because some NAT devices have an upper limit on how quickly bindings will be allocated. Section 5 restricts the rate at which clients may begin new STUN transactions.

4.6. Binding Lifetime Discovery

STUN can also be used to probe the lifetimes of the bindings created by the NAT. Such tests are sensitive to prior state on the NAT. For many NAT devices, an absolute refresh interval cannot be determined; bindings might be closed more quickly under heavy load or might not behave as the tests suggest. For this reason, applications that require reliable bindings must send keepalives as frequently as required by all NAT devices that will be encountered. Suggested refresh intervals are outside the scope of this document. [RFC5245] and OUTBOUND [RFC5626] have suggested refresh intervals.

Determining the binding lifetime relies on two separate source ports being used to send STUN Binding Requests to the STUN server. The general approach is that the client uses a source port X to send a single Binding Request. After a period of time during which source port X is not used, the client uses a second source port Y to send a Binding Request to the STUN server that indicates the response should be sent to the binding established to port X. If the binding for port X has timed out, that response will not be received. By varying the time between the original Binding Request sent from X and the subsequent request sent from Y, the client can determine the binding lifetime.

To determine the binding lifetime, the client first sends a Binding Request to the server from a particular source port, X. This creates a binding in the NAT. The response from the server contains a MAPPED-ADDRESS attribute, providing the public address and port on the NAT. Call this Pa and Pp, respectively. The client then starts a timer with a value of T seconds. When this timer fires, the client sends another Binding Request to the server, using the same destination address and port, but from a different source port, Y. This request contains an RESPONSE-PORT attribute, set to Pp, to request the response be delivered to (Pa, Pp). This will create a new binding on the NAT, and cause the STUN server to send a Binding Response that would match the old binding, (Pa, Pp), if it still exists. If the client receives the Binding Response on port X, it knows that the binding has not expired. If the client receives the Binding Response on port Y (which is possible if the old binding expired, and the NAT allocated the same public address and port to the new binding), or receives no response at all, it knows that the binding has expired.

Because some NATs only refresh bindings when outbound traffic is sent, the client must resend a binding request from the original source port before beginning a second test with a different value of T. The client can find the value of the binding lifetime by doing a binary search through T, arriving eventually at the value where the response is not received for any timer greater than T, but is received for any timer less than T. Note also that the binding refresh behavior (outbound only or all traffic) can be determined by sending multiple Binding Requests from port Y without refreshes from the original source port X.

This discovery process takes quite a bit of time and is something that will typically be run in the background on a device once it boots.

It is possible that the client can get inconsistent results each time this process is run. For example, if the NAT should reboot, or be reset for some reason, the process may discover a lifetime that is shorter than the actual one. Binding lifetime may also be dependent on the traffic load on the NAT. For this reason, implementations are encouraged to run the test numerous times and be prepared to get inconsistent results.

Like the other diagnostics, this test is inherently unstable. In particular, an overloaded NAT might reduce binding lifetime to shed load. A client might find this diagnostic useful at startup, for example, setting the initial keepalive interval on its connection to the server to 10 seconds while beginning this check. After determining the current lifetime, the keepalive interval used by the

connection to the server can be set to this appropriate value. Subsequent checks of the binding lifetime can then be performed using the keepalives in the server connection. The STUN Keepalive Usage [RFC5626] provides a response that confirms the connection is open and allows the client to check that its mapped address has not changed. As that provides both the keepalive action and diagnostic that it is working, it should be preferred over any attempt to characterize the connection by a secondary technique.

5. Client Behavior

Unless otherwise specified here, all procedures for preparing, sending, and processing messages as described in the STUN Binding Usage [RFC5389] are followed.

As support for RESPONSE-PORT is optional, a client MUST be prepared to receive a 420 (Unknown Attribute) error to requests that include RESPONSE-PORT. Support for OTHER-ADDRESS and CHANGE-REQUEST is optional, but MUST be supported by servers advertised via SRV, as described below. This is to allow the use of PADDING and RESPONSE-PORT in applications where servers do not have multiple IP addresses. Clients MUST be prepared to receive a 420 for requests that include CHANGE-REQUEST when OTHER-ADDRESS was not received in Binding Response messages from the server.

If an application makes use of the NAT Behavior Discovery STUN usage by multiplexing it in a flow with application traffic, a FINGERPRINT attribute SHOULD be included unless it is always possible to distinguish a STUN message from an application message based on their header.

When PADDING is used, it SHOULD be equal to the MTU of the outgoing interface.

Clients SHOULD ignore an ALTERNATE-SERVER attribute in a response unless they are using authentication with a provider of STUN servers that is aware of the topology requirements of the tests being performed.

A client SHOULD NOT generate more than ten new STUN transactions per second and SHOULD pace them such that the retransmission timeouts (RTOs) do not synchronize the retransmissions of each transaction.

5.1. Discovery

Unless the user or application is aware of the transport address of a STUN server supporting the NAT Behavior Discovery usage through other means, a client is configured with the domain name of the provider of

the STUN servers. The domain is resolved to a transport address using SRV procedures [RFC2782]. The mechanism for configuring the client with the domain name of the STUN servers or of acquiring a specific transport address is out of scope for this document.

For the Behavior Discovery usage, the service name is "stun-behavior" for UDP and TCP. The service name is "stun-behaviors" for TLS over TCP. Only "tcp" is defined as a protocol for "stun-behaviors". Other aspects of handling failures and default ports are followed as described in STUN [RFC5389].

5.2. Security

Servers MAY require authentication before allowing a client to make use of its services. The method for obtaining these credentials, should the server require them, is outside the scope of this usage. Presumably, the administrator or application relying on this usage should have its own method for obtaining credentials. If the client receives a 401 (Unauthorized) Response to a Request, then it must either acquire the appropriate credential from the application before retrying or report a permanent failure. Procedures for encoding the MESSAGE-INTEGRITY attribute for a request are described in STUN [RFC5389].

6. Server Behavior

Unless otherwise specified here, all procedures for preparing, sending, and processing messages as described for the STUN Binding Usage of STUN [RFC5389] are followed.

A server implementing the NAT Behavior Discovery usage SHOULD be configured with two separate IP addresses on the public Internet. On startup, the server SHOULD allocate a pair of ports for each of the UDP, TCP, and TCP/TLS transport protocols, such that it can send and receive datagrams using the same ports on each IP address (normally a wildcard binding accomplishes this). TCP and TCP/TLS MUST use different ports. If a server cannot allocate the same ports on two different IP address, then it MUST NOT include an OTHER-ADDRESS attribute in any Response and MUST respond with a 420 (Unknown Attribute) to any Request with a CHANGE-REQUEST attribute. A server with only one IP address MUST NOT be advertised using the SRV service name "stun-behavior" or "stun-behaviors".

6.1. Preparing the Response

After performing all authentication and verification steps, the server begins processing specific to this Usage if the Binding Request contains any request attributes defined in this document:

RESPONSE-PORT, CHANGE-REQUEST, or PADDING. If the Binding Request does not contain any attributes from this document, OTHER-ADDRESS and RESPONSE-ORIGIN are still included in the Binding Response.

The server MUST include both MAPPED-ADDRESS and XOR-MAPPED-ADDRESS in its Response.

If the Request contains the CHANGE-REQUEST attribute and the server does not have an alternate address and port as described above, the server MUST generate an error response of type 420.

The source address and port of the Binding Response depend on the value of the CHANGE-REQUEST attribute and on the address and port on which the Binding Request was received; this is summarized in Table 1.

Let A1 and A2 be the two IP addresses used by the server, and P1 and P2 be the ports used by the server. Let Da represent the destination IP address of the Binding Request (which will be either A1 or A2), and Dp represent the destination port of the Binding Request (which will be either P1 or P2). Let Ca represent the other address, so that if Da is A1, Ca is A2. If Da is A2, Ca is A1. Similarly, let Cp represent the other port, so that if Dp is P1, Cp is P2. If Dp is P2, Cp is P1. If the "change port" flag was set in the CHANGE-REQUEST attribute of the Binding Request, and the "change IP" flag was not set, the source IP address of the Binding Response MUST be Da and the source port of the Binding Response MUST be Cp. If the "change IP" flag was set in the Binding Request, and the "change port" flag was not set, the source IP address of the Binding Response MUST be Ca and the source port of the Binding Response MUST be Dp. When both flags are set, the source IP address of the Binding Response MUST be Ca and the source port of the Binding Response MUST be Cp. If neither flag is set, or if the CHANGE-REQUEST attribute is absent entirely, the source IP address of the Binding Response MUST be Da and the source port of the Binding Response MUST be Dp.

Flags	Source Address	Source Port	OTHER-ADDRESS
none	Da	Dp	Ca:Cp
Change IP	Ca	Dp	Ca:Cp
Change port	Da	Cp	Ca:Cp
Change IP and Change port	Ca	Cp	Ca:Cp

Table 1: Impact of Flags on Packet Source and OTHER-ADDRESS

The server MUST add a RESPONSE-ORIGIN attribute to the Binding Response, containing the source address and port used to send the Binding Response.

If the server supports an alternate address and port, the server MUST add an OTHER-ADDRESS attribute to the Binding Response. This contains the source IP address and port that would be used if the client had set the "change IP" and "change port" flags in the Binding Request. As summarized in Table 1, these are Ca and Cp, respectively, regardless of the value of the CHANGE-REQUEST flags.

If the Request contained a PADDING attribute, PADDING MUST be included in the Binding Response. The server SHOULD use a length of PADDING equal to the MTU on the outgoing interface, rounded up to an even multiple of four bytes. If the Request also contains the RESPONSE-PORT attribute the server MUST return an error response of type 400.

Following that, the server completes the remainder of the processing from STUN [RFC5389]. If authentication is being required, the server MUST include a MESSAGE-INTEGRITY and associated attributes as appropriate. A FINGERPRINT attribute is only required if the STUN messages are being multiplexed with application traffic that requires use of a FINGERPRINT to distinguish STUN messages.

An ALTERNATE-SERVER attribute MUST NOT be included with any other attribute defined in this specification.

When the server sends the Response, it is sent from the source address as determined above and to the source address of the Request. If RESPONSE-PORT is present, the server sends the response to that port instead of the originating port.

7. New Attributes

This document defines several STUN attributes that are required for NAT Behavior Discovery. These attributes are all used only with Binding Requests and Binding Responses. CHANGE-REQUEST was originally defined in RFC 3489 [RFC3489] but is redefined here as that document is obsoleted by [RFC5389].

```
  Comprehension-required range (0x0000-0x7FFF):
    0x0003: CHANGE-REQUEST
    0x0026: PADDING
    0x0027: RESPONSE-PORT
```

```
   Comprehension-optional range (0x8000-0xFFFF):
     0x802b: RESPONSE-ORIGIN
     0x802c: OTHER-ADDRESS
```

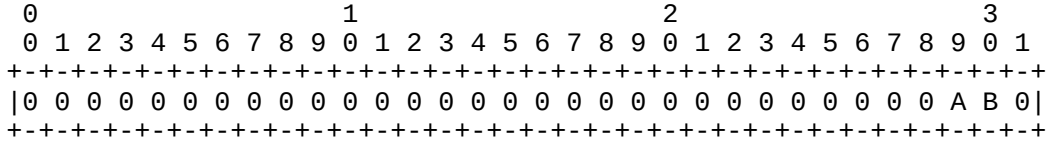
7.1. Representing Transport Addresses

Whenever an attribute contains a transport IP address and port, it has the same format as MAPPED-ADDRESS. Similarly, the XOR-attributes have the same format as XOR-MAPPED-ADDRESS [RFC5389].

7.2. CHANGE-REQUEST

The CHANGE-REQUEST attribute contains two flags to control the IP address and port that the server uses to send the response. These flags are called the "change IP" and "change port" flags. The CHANGE-REQUEST attribute is allowed only in the Binding Request. The "change IP" and "change port" flags are useful for determining the current filtering behavior of a NAT. They instruct the server to send the Binding Responses from the alternate source IP address and/or alternate port. The CHANGE-REQUEST attribute is optional in the Binding Request.

The attribute is 32 bits long, although only two bits (A and B) are used:



The meanings of the flags are:

A: This is the "change IP" flag. If true, it requests the server to send the Binding Response with a different IP address than the one the Binding Request was received on.

B: This is the "change port" flag. If true, it requests the server to send the Binding Response with a different port than the one the Binding Request was received on.

7.3. RESPONSE-ORIGIN

The RESPONSE-ORIGIN attribute is inserted by the server and indicates the source IP address and port the response was sent from. It is useful for detecting double NAT configurations. It is only present in Binding Responses.

7.4. OTHER-ADDRESS

The OTHER-ADDRESS attribute is used in Binding Responses. It informs the client of the source IP address and port that would be used if the client requested the "change IP" and "change port" behavior. OTHER-ADDRESS MUST NOT be inserted into a Binding Response unless the server has a second IP address.

OTHER-ADDRESS uses the same attribute number as CHANGED-ADDRESS from RFC 3489 [RFC3489] because it is simply a new name with the same semantics as CHANGED-ADDRESS. It has been renamed to more clearly indicate its function.

7.5. RESPONSE-PORT

The RESPONSE-PORT attribute contains a port. The RESPONSE-PORT attribute can be present in the Binding Request and indicates which port the Binding Response will be sent to. For servers which support the RESPONSE-PORT attribute, the Binding Response MUST be transmitted to the source IP address of the Binding Request and the port contained in RESPONSE-PORT. It is used in tests such as Section 4.6. When not present, the server sends the Binding Response to the source IP address and port of the Binding Request. The server MUST NOT process a request containing a RESPONSE-PORT and a PADDING attribute. The RESPONSE-PORT attribute is optional in the Binding Request. Server support for RESPONSE-PORT is optional.

RESPONSE-PORT is a 16-bit unsigned integer in network byte order followed by 2 bytes of padding. Allowable values of RESPONSE-PORT are 0-65536.

7.6. PADDING

The PADDING attribute allows for the entire message to be padded to force the STUN message to be divided into IP fragments. PADDING consists entirely of a free-form string, the value of which does not matter. PADDING can be used in either Binding Requests or Binding Responses.

PADDING MUST NOT be longer than the length that brings the total IP datagram size to 64K. It SHOULD be equal in length to the MTU of the outgoing interface, rounded up to an even multiple of four bytes. Because STUN messages with PADDING are intended to test the behavior of UDP fragments, they are an exception to the usual rule that STUN messages be less than the MTU of the path.

8. IAB Considerations

The IAB has studied the problem of "Unilateral Self-Address Fixing" (UNSAF), which is the general process by which a client attempts to determine its address in another realm on the other side of a NAT through a collaborative protocol reflection mechanism [RFC3424]. The STUN NAT Behavior Discovery usage is an example of a protocol that performs this type of function. The IAB has mandated that any protocols developed for this purpose document a specific set of considerations. This section meets those requirements.

8.1. Problem Definition

From RFC 3424 [RFC3424], any UNSAF proposal must provide:

> Precise definition of a specific, limited-scope problem that is to be solved with the UNSAF proposal. A short term fix should not be generalized to solve other problems. Such generalizations lead to the prolonged dependence on and usage of the supposed short term fix -- meaning that it is no longer accurate to call it "short term".

The specific problem being solved by the STUN NAT Behavior Discovery usage is for a client, which may be located behind a NAT of any type, to determine the instantaneous characteristics of that NAT. This determination allows either the diagnosis of the cause of problems experienced by that or other applications or the modification of an application’s behavior based on the current behavior of the NAT and an appropriate statistical model of the behavior required for the application to succeed.

8.2. Exit Strategy

From [RFC3424], any UNSAF proposal must provide:

> Description of an exit strategy/transition plan. The better short term fixes are the ones that will naturally see less and less use as the appropriate technology is deployed.

The STUN NAT Behavior Discovery usage does not itself provide an exit strategy for v4 NATs. At the time of this writing, it appears some sort of NAT will be necessary between v6 clients and v4 servers, but this specification will not be necessary with those v6-to-v4 NATs because the IETF is planning to adequately describe their operation. This specification will be of no interest for v6-to-v6 connectivity.

8.3. Brittleness Introduced by STUN NAT Behavior Discovery

From [RFC3424], any UNSAF proposal must provide:

> Discussion of specific issues that may render systems more "brittle". For example, approaches that involve using data at multiple network layers create more dependencies, increase debugging challenges, and make it harder to transition.

The STUN NAT Behavior Discovery usage allows a client to determine the current behavior of a NAT. This information can be quite useful to a developer or network administrator outside of an application, and as such can be used to diagnose the brittleness induced in another application. When used within an application itself, STUN NAT Behavior Discovery allows the application to adjust its behavior according to the current behavior of the NAT. This document is experimental because the extent to which brittleness is introduced to an application relying on the Behavior Discovery usage is unclear and must be carefully evaluated by the designers of the protocol making use of it. The experimental test for this protocol is essentially determining whether an application can be made less brittle through the use of behavior-discovery information than it would be if attempted to make use of the network without any awareness of the NATs its traffic must pass through.

8.4. Requirements for a Long-Term Solution

From [RFC3424], any UNSAF proposal must provide:

> Identify requirements for longer-term, sound technical solutions -- contribute to the process of finding the right longer-term solution.

As long as v4 NATs are present, means of adapting to their presence will be required. As described above, well-behaved v6 to v4 NATs and direct v6 to v6 connections will not require behavior characterization.

8.5. Issues with Existing NAPT Boxes

From [RFC3424], any UNSAF proposal must provide:

> Discussion of the impact of the noted practical issues with existing deployed NATs and experience reports.

This usage provides a set of generic attributes that can be assembled to test many types of NAT behavior. While tests for the most commonly known NAT box behaviors are described, the BEHAVE mailing

list regularly has descriptions of new behaviors, some of which may not be readily detected using the tests described herein. However, the techniques described in this usage can be assembled in different combinations to test NAT behaviors not now known or envisioned.

9. IANA Considerations

9.1. STUN Attribute Registry

This specification defines several new STUN attributes. IANA has added these new protocol elements to the "STUN Attributes" registry.

0x0003: CHANGE-REQUEST
0x0027: RESPONSE-PORT
0x0026: PADDING
0x8027: CACHE-TIMEOUT
0x802b: RESPONSE-ORIGIN
0x802c: OTHER-ADDRESS

9.2. Port Numbers and SRV Registry

By default, the STUN NAT Behavior Discovery usage runs on the same ports as STUN: 3478 over UDP and TCP, and 5349 for TCP over TLS. However, the Behavior Discovery usage has its own set of Service Record (SRV) names: "stun-behavior" for UDP and TCP, and "stun-behaviors" for TLS. Either the SRV procedures or the ALTERNATE-SERVER procedures, subject to the recommendations of Section 5, can be used to run Behavior Discovery on a different port.

This specification defines the "stun-behavior" and "stun-behaviors" SRV service names. "stun-behavior" may be used with SRV protocol specifiers "udp" and "tcp". "stun-behaviors" may only be specified with "tcp". Thus, the allowable SRV queries are:

_stun-behavior._udp	UDP
_stun-behavior._tcp	TCP
_stun-behaviors._tcp	TLS over TCP

10. Security Considerations

This usage inherits the security considerations of STUN [RFC5389]. This usage adds several new attributes; security considerations for those are detailed here.

OTHER-ADDRESS does not permit any new attacks; it provides another place where an attacker can impersonate a STUN server but it is not an interesting attack. An attacker positioned where it can compromise the Binding Response can completely hide the STUN server from the client.

- Requests containing both RESPONSE-PORT and PADDING are rejected by the server. This prevents an amplification attack that is targeted at the originating address.

The only attack possible with the PADDING attribute is to have a large padding length that could cause a server to allocate a large amount of memory. As servers will ignore any padding length greater than 64K so the scope of this attack is limited. In general, servers should not allocate more memory than the size of the received datagram. This attack would only affect non-compliant implementations.

RESPONSE-ORIGIN and RESPONSE-PORT do not provide any additional attacks.

11. Acknowledgements

The authors would like to thank the authors of the original STUN specification [RFC3489] from which many of the ideas, attributes, and description in this document originated. Thanks to Dan Wing, Cullen Jennings, and Magnus Westerlund for detailed comments.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

12.2. Informative References

[RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

Authors' Addresses

Derek C. MacDonald
Skype
Palo Alto, CA
USA

EMail: derek.macdonald@gmail.com

Bruce B. Lowekamp
Skype
Palo Alto, CA
USA

EMail: bbl@lowekamp.net