Network Working Group
Internet-Draft
Expires: February 28, 2013

Y. Gu
Huawei
M. Shore
No Mountain Software
S. Sivakumar
Cisco Systems
August 27, 2012

A Framework and Problem Statement for Flow-associated Middlebox State Migration
draft-gu-statemigration-framework-02

Abstract

This document presents an initial framework and discussion of the problem of transferring middlebox (for example, firewall or NAT) flow-coupled state from one middlebox to another while the flow is still active. This has most recently come up in the context of virtual machine (VM) migration between hypervisors, but it is a problem that has appeared in other situations, as well. We present some of the parameters of the problem, define some language for discussing the problem, and begin to identify a path forward for addressing it.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 28, 2013.

Copyright Notice

Table of Contents

1. Introduction

An end-to-end network flow typically traverses one or more "middlebox," which may retain state about the flow. These include, for example, firewalls, NATs, traffic optimizers, and similar. The flow-associated state is usually instantiated through a combination of traffic inspection and broad policies, but may also be created by the use of an explicit request or signaling mechanism.

When an endpoint changes its point of attachment to a network, it retains its IP address, and the standard 5-tuple used to describe a flow (source and destination addresses, source and destination ports, protocol) stay the same. Because of this it is possible to move existing middlebox state containing these elements.

The problem of how to handle transfering flow-associated middlebox state when one flow endpoint moves is not a new one, but with some exceptions it remains largely unaddressed. For example, situations in which one endpoint or another "move" (we define what it means to move an endpoint in more detail in Section 5) include mobile IP [RFC5944], failover in a high-availability deployment, and VM (virtual machine) migration. Related problems include multihomed endpoints in SCTP and load balancing.

In this document we establish terminology (Section 2), describe the problem, and lay out the components of the problem that would need to be addressed in a solution.

2. Terminology

flow: "Traffic flow" is defined in [RFC2722] as an artificial logical equivalent of a call or connection. It is delimited by a start and a stop time.

middlebox: A middlebox was defined in [RFC3234] as "any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and a destination host." RFC 3234 provides an older but excellent and still-relevant taxonomy of middlebox types.

move: When we talk about an endpoint "moving" what we are describing is the endpoint changing its point of attachment to the network. For the purpose of this discussion we assume that it retains the same IP address after the move that it had before the move.

policy: See Section 4.2

3. Goals

The problem we are interested in solving is the question of how to keep longer-lived network flows "alive" when an endpoint’s point of attachment to a network changes. The particular piece of this we intend to address is how to move the middlebox (in this case, firewall or NAT) state associated with a network flow to new middleboxes.

4. Middlebox state

4.1. What state is associated with a flow on a middlebox?

To date, we haven't been able to find a normative definition of the term 'state' in IETF documents. More generally it tends to be considered to be a set of observable properties associated with an object. This is (largely) distinct from automata theory, in which "state" refers to the condition of an object (or automaton). The observable things which might be associated by a middlebox with a network flow are described below.

Transport-layer middleboxes which keep flow-associated state through the duration of the flow typically keep, at a minimum, the standard IP 5-tuple:

{s_addr, d_addr, s_port, d_port, protocol}

where

s_addr is the source address

d_addr is the destination address

s_port is the source port

d_port is the destination port

protocol is the IP protocol (TCP, UDP, SCTP, RSVP, etc.)

Other data elements often associated with a network flow include timers.

Over the lifetime of a flow, it is not expected that elements of the standard 5-tuple will change, but there may be other pieces of state, such as timers, or data extracted from stateful inspection, which may be expected to change before a flow terminates.

As mentioned above, when an endpoint "moves" it retains its IP address(es) and the sockaddr information associated with a flow on an endpoint does not change.

Middlebox state is almost always associated with a specific interface (rather than the interface being an attribute of the flow). Some "stateful inspection" firewalls may keep state from higher layers in the networking stack: everything from TCP sequence numbers to entire SIP dialogues.

Note that the state associated with a flow may be left up when the flow is torn down in some implementations, such as those NATs that put the state on an activity-based timer as an efficiency mechanism, to avoid reinstantiating state should a new flow be created which shares the attributes of the flow which just ended. This is often the case with HTTP, for example.

It should also be noted that it is possible that a given bidirectional network flow (say, TCP) may have each flow (to and from its peer) follow different routes, commonly referred to as "asymmetric routing." When an endpoint moves, it is possible that

- o both flows traverse the same middlebox before the move and after the move,
- o both flows traverse the same middlebox before the move and different middleboxes after the move,
- o both flows traverse different middleboxes before the move but the same middlebox after the move, or
- o both flows traverse different middleboxes before the move and different middleboxes after the move

4.2. State vs policy

We would like to draw a clear distinction between state and policy. 'Policy' is a set of statements that define how traffic (in this case) is to be treated by the middlebox. In some sense policy is a description of what state should be applied to a network flow; that is to say, state includes the instantiation of policy. When a flow first arrives at a middlebox, it consults its policy to determine what state (if any) is to be created and then associated with that flow

As a general rule of thumb, policy is provisioned while state represents run-time responses to environmental conditions (in this case, network flows). Because policy is provisioned and because we assume that the middleboxes between which state would be migrated are under the administrative control of the same organization, we will make another assumption that there is consistent policy configured across middleboxes. We are aware that this is not always a correct assumption.

Note that implicit in this description is the notion of policy definition having an administrative scope. That is to say, there is an assumption that state must only be migrated between middleboxes in the same administrative policy domain. There are several risks

associated with migrating state between middleboxes in different administrative domains, prominent among which is the possibility of installing local state on the "new" middlebox which violates its policy. We feel that migrating state between middleboxes in different administrative policy domains should be considered out of scope for the time being.

4.3. Mechanisms for instantiating middlebox state

State is created on middleboxes using a small number of mechanisms, sometimes in combination.

The most common means by which middlebox state is created is that the middlebox examines traffic and compares it against its own policies, which have typically been configured or provisioned by a systems or network administrator but in very simple cases can come preprovisioned, for example on commodity consumer equipment. It then creates middlebox state, in the form of a firewall pinhole, a NAT table mapping, QoS table entry, etc.

Another means is through explicit request. An endpoint or its proxy sends a request for resources (again, firewall pinhole, NAT table mapping, and so on) to the middlebox using some sort of "signaling" protocol to request the resource. The middlebox compares the request to its policy and grants or denies the request based on that policy. Examples of explicit request include RSVP [RFC2205], midcom [RFC3303], TURN [RFC5766], and the work being done by the IETF pcp [1] working group.

It is worth mention that there are mechanisms that are essentially hybrids of the previous two approaches, using expected effects of sending traffic across a middlebox to trigger hoped-for state instantiation. STUN [RFC5389] is probably the best-known example of this.

5. "Moving" endpoints

Moving an endpoint, in the context of this internet draft, refers to changing its point of attachment to a network. Doing so may cause traffic to cross different middleboxes from the ones the traffic traversed when the middlebox state was created.

5.1. A few words about addresses

One question that comes up from time to time in discussions of VM migration is whether or not the IP address will change as a result of the migration. We believe that this is out of scope for the time being, not the least because host operating system support is potentially difficult. If our goal is to keep a given network flow up and alive during a migration, not only would the endpoint operating system need to be aware that its address has changed, it would also need to to be able to signal the other end of the flow, which would have to respond by modifying open sockets' sockaddrs, etc. There are also some obvious security problems that would need to be addressed.

5.2. Scenarios

In this section we introduce a few scenarios. We believe the problem characteristics are fundamentally the same in these scenarios and that what we're describing is a general problem.

5.2.1. Virtual machine migration

The live migration (i.e. the VM appears to remain "up" and available during the migration - that is to say, TCP or other connection-oriented flows are not dropped) of virtual machines between hypervisors in the same data center has been established practice for several years now, but there's been a move towards live migration of VMs between geographically disparate data centers (see, for example this collaboration [2] between Cisco and VMWare). This provides the ability to perform data center maintenance without downtime, data center migration or consolidation, data center expansion, and workload balancing. There is a compelling use case for VM migration.

5.2.2. SCTP NAT

The SCTP [RFC4960] protocol supports multihomed endpoints. Any NAT that is port-aware (and these days it is nearly all of them) will need to have SCTP support in order to be able to handle extracting the port numbers even for flows that are single-homed on each end. This provides a mechanism for transparent failover when one path taken by the network flow fails (see section 6.4 in [RFC4960]

The upshot of this is that if a NAT is maintaining state related to a flow on the primary path and the primary path fails, that state may need to be transferred to the NAT being traversed by the secondary path.

This problem is being addressed in the IETF behave [3] working group.

5.2.3. High availability, and failover

"High-availability" commonly suggests failover as a mechanism to guarantee uninterrupted (or minimally interrupted) services. When a failure is detected services are shifted onto a secondary server. Note that this shift can be implemented through VM migration, as well as having the services brought up on a new system image.

Because outages are sometimes caused by site failures, failover can take place across geographically disparate sites. This introduces the likelihood of the flow now traversing a very different network path and a new set of middleboxes.

6. "Directionality"

One of the questions that comes up when considering an overall architecture to solve this set of problems is who initiates the state migration and how the data "flow" from place to place.

One approach is to have the middleboxes communicate directly with each other. In this case having all middleboxes poll all other middleboxes for copies of their state seems wasteful and inefficient, suggesting that communication between middleboxes would need a specific trigger. The "old" middlebox could send its state to the "new" middlebox or the new middlebox could send a request to the old middlebox for a copy of its state. In either case one middlebox would need to know the location of the other and be able to communicate with it (both parties would need to authenticate to each other). Note that if a catastrophic network event caused the old middlebox to become unreachable, it would be impossible to successfully query it for its state. [Note that this approach was considered for SCTP NAT traversal and discarded as impossible, since there was no way for one NAT to know about other NATs.]

Another approach is to have some controlling entity involved, either mediating communication between middleboxes or directing communication between middleboxes. In a VM migration scenario, a VM manager, or a network manager communicating with a VM manager, is an obvious candidate. As described in Section 4.2, the migration must stay within an administrative policy boundary, which may eliminate the need for multiple mediators.

The orthogonal question to whether or not there’s a mediating entity is who initiates the communication - does the old middlebox respond to a catastrophic event by dumping state before shutting down (not always possible, obviously) or is it polled by a mediating device or a new middlebox? Another possibility is to periodically transfer incremental information so that a non-recoverable error can save most of the flows, if not all.

7. Problems

The problems that must be solved in order to move middlebox state along with a moving endpoint include:

- Recognizing when an endpoint has moved
- Locating middleboxes along the original path
- Locating middleboxes along the new path
- Getting a copy of state from middleboxes along the old path
- Installing that state in middleboxes along the new path

7.1. Recognizing when an endpoint has moved

As touched upon in Section 5.2, there are various circumstances that could cause an endpoint to change its point of attachment to a network. They fall into two broad categories: planned and unplanned.

In the planned case, some entity knows that an endpoint is about to move and the move can happen in a controlled fashion. There may be time to send network queries, learn topology, and gather state.

The unplanned case is typically a response to the failure of some element in the network. A monitoring heartbeat is missed, a connection times out, or some other indication of catastraphic failure is received by an endpoint or by a monitoring service. Not only does this interfere with the notion of an organized transfer from one path to the new one, it also means that there may be cases where the old middlebox is not reachable and it’s not possible to query its state.

7.2. Topology discovery

Somehow or other the state migration mechanism needs to be able to locate and communicate with both the middleboxes on the old path and the middleboxes on the new path. This is not a trivial problem; IP was designed to have the network itself be largely opaque to endpoints, and very often systems and network administrators prefer not to expose network topology, feeling that it would introduce security threats.

There are several options, including configuration, discovery, and notification. In configuration, someone with knowledge of the network topology would be able to construct a table describing middleboxes associated with certain routes. In discovery, a network

mechanism would be used to query for the middleboxes along a path, similar to traceroute or to a PATH message in RSVP [RFC2205].

A configuration mechanism would have the disadvantage of being not particularly responsive to changes in the network, as well as being somewhat error-prone. However, it would not involve inventing a new network mechanism or requiring changes on every participating middlebox (although the state migration mechanism itself would nearly certainly require changes).

[Note that an architecture that had the middlebox copying its own state out to some third party would almost certainly have to be configuration-base.]

A discovery-based approach would require putting new software on every middlebox, an approach that is intuitively unappealing and that has been repeatedly shown to inhibit adoption of newer technologies. There is no such thing as incremental deployment using this approach. It also introduces security problems, since without the appropriate protections it would allow attackers to probe and discover not just network topology but specifically the location of security devices/ middleboxes in a given network. On the other hand it’s robust against configuration errors and highly responsive to changes in the underlying network.

A third option, notification, relies on a middlebox announcing its presence to the network, typically using anycast or broadcast. This also requires changes to both the middlebox and a controlling entity, and a an announcement/notification protocol. It has the advantage of being responsive to new middleboxes coming up in the network, although a mechanism (such as a heartbeat) would be needed to detect outages and drops.

The primary security consideration in a notification scenario is that the network must be tightly controlled to prevent announcements from being eavesdropped upon by adversaries.

7.3. Copying state from a middlebox

Another problem to be solved is the one of copying state from a middlebox, encoding it, and transferring it over the network.

It may be the case that the middleboxes are from different manufacturers/vendors, and so the problem of representing the state we wish to transfer includes the question of presenting it in a vendor-neutral format, including both state semantics and state syntax.

A somewhat more challenging aspect of this problem is how to transport the encoded state. For one thing, it may be that the event that triggered the endpoint migration has also rendered the middlebox in question unreachable. For another, what sort of load this imposes on the middlebox depends, among other things, on the "directionality" of the state migration. It may be that an external device, such as a session controller, a hypervisor, or another middlebox queries the old middlebox for a copy of its state. In high-availability scenarios the middlebox may end up "pushing" copies of its state out to some controlling or intermediate entity, such as a hypervisor.

Among the transport characteristics that need to be considered is reliability, and being able to recognize when a copy of the source middlebox state has not been transferred correctly, whether it’s because it’s incomplete, damaged, or inauthentic.

7.4. Installing state on the new middlebox

The problem of installing state on the new middlebox is closely related to the one of copying state from the old middlebox. In both cases we’re facing the problems of representation and encoding, a transport protocol to/from the middlebox, and questions about reachability.

Reliability is a question here, as well, with the additional concern, beyond what is described in the previous section, of whther or not the state is installed correctly on the new middlebox. Issues that could interfere with installation include resource limitations, and authority/authorization.

8. Security Considerations

Any time we introduce new mechanisms to query and manipulate middleboxes, we also introduce potentially very serious security exposures.

In this case, because we're planning on discovering the location of middleboxes, querying the middleboxes for their state, and installing state on middleboxes, we face a very broad range indeed of potential threats.

Network and systems administrators typically want to conceal network topology from outsiders, and it may be necessary to use authenticated discovery (packet filtering may be adequate for some deployments but not all). This introduces problems around credentials management and keying for participants, and may suggest that we would want to minimize the number of network elements talking with one another.

Cleary the ability to copy data from a middlebox introduces the ability to discovery yet more network topology, and in particular to identify specific firewall pinholes and NAT table mappings, and their associated state.

Similarly, the ability to install state on a middlebox can introduce both Denial of Service (DoS) vulnerabilities but also the ability of an attacker to penetrate a middlebox, or to disable it completely.

In all cases, protections must be designed with sensitivity to performance, since middleboxes often are processing very heavy traffic loads. This means keeping an eye on cryptographic processing demands, key and other credentials management, etc.

9. IANA Considerations

This document has no actions for IANA.

10. Acknowledgments

Many thanks to David Black for his careful review and suggestions for improvements.

11. Informative References

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2722] Brownlee, N., Mills, C., and G. Ruth, "Traffic Flow Measurement: Architecture", RFC 2722, October 1999.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[RFC4067] Loughney, J., Nakhjiri, M., Perkins, C., and R. Koodli, "Context Transfer Protocol (CXTP)", RFC 4067, July 2005.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

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

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC5944] Perkins, C., "IP Mobility Support for IPv4, Revised", RFC 5944, November 2010.

[1] <http://datatracker.ietf.org/wg/pcp/charter/>

[2] <http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns836/white_paper_c11-557822.pdf>

[3] <http://datatracker.ietf.org/wg/behave/>

Appendix A. On the applicability of the Context Transfer Protocol

In this section we examine the applicability of the Context Transfer Protocol [RFC4067] to the state migration problem, given the problems outlined in Section 7. In Section 7, we identify the following components of the overall state migration problem:

- Recognizing when an endpoint has moved
- Locating middleboxes along the original path
- Locating middleboxes along the new path
- Getting a copy of state from middleboxes along the old path
- Installing that state in middleboxes along the new path

A.1. Topology awareness

The Context Transfer Protocol was designed to support node mobility -- to minimize disruptions when a mobile node attaches to a new access router. In a typical scenario, when a mobile node moves from one access router to another, CXTP provides a means to move associated state (or context) to the new access router to which the node becomes attached.

In the CXTP scenario, the mobile node "knows" that the access router is there and has direct communication with it, by virtue of the underlying network mobility mechanisms. A context transfer may be initiated by the mobile node when it "knows" that it will be attaching to a new access router, or it may be initiated by the existing access router when it receives a link-layer trigger. Alternatively, a context transfer may be initiated by the new access router when it receives a link-layer trigger. In short, the context transfer request is generated by a first party in the network, either the mobile node itself or one of its access routers.

This contrasts rather starkly with the usual middlebox scenario, where the middlebox is typically invisible to the endpoint (the mobile node analogue). A mobile node has an explicit relationship with an access router; a network endpoint has no such relationship with a firewall or NAT, except in those cases in which the firewall or NAT is doing double-duty as a proxy.

Topology awareness has been one of the most persistent and difficult problems associated with middlebox communication issues. In the CXTP case topology awareness is pre-existing in the network and the relationship between the mobile node and the access router.

A.2. Triggers

The question of the triggers initiating a context transfer or state migration is very closely tied to the question of topology awareness, since in the CXTP case the mobile node "knows" the access router is there and has an explicit relationship with it, while in the state migration case the middlebox is opaque to the endpoint.

The mechanisms underlying a mobile node attach/detach differ significantly from those underlying, say, a virtual machine migration. At the most basic level, a mobile node knows that it is moving between access routers. A virtual machine typically does not know that it’s being moved - the VM migration is triggered by a third party and is opaque to the VM itself, since its own state is maintained intact across the migration. A network access device may detect a change, but it will not have knowledge of the other (previous) middlebox nor will it be able to request that information from the migrated VM, since the VM itself will not know whether or not there were middleboxes present, or where they were, as described in the previous section.

A.3. Copying state

CXTP has been designed to transfer state between a source access router and a destination access router -- that is to say, they must know about each other, know that a given mobile node is associated with the other access router, and have a network path between the two access routers.

That is not the case when migrating virtual machines. The network element which triggers a VM migration does not necessarily have network topology awareness and does not have sufficient information to be able to request a migration of associated state.

That said, CXTP looks highly suitable for actually transferring the middlebox state, once the topology/ middlebox discovery problems are solved. Security issues would need an extra level of scrutiny, not only because, as described in [RFC4067], the threats in a handover were not well understood at the time the document was published, but also because the network elements involved are different and the relationships among those network elements are different. Having a third party (the element requesting the VM migration) request a migration of network middlebox state requires different security properties from having a network element (a mobile node or its access routers) request a context transfer on its own behalf.

A.4. Conclusion

Based on the previous discussion we believe that CXTP may be directly useful for the actual transfer of middlebox state but that it does not address some core problems which would need to be solved in order to successfully migrate that state. These problems are topology discovery (i.e. locating the correct middleboxes), and generating triggers.

Authors' Addresses

Yingjie Gu
Huawei

Phone: +86-25-56624760
Fax: +86-25-56624702
Email: guyingjie@huawei.com

Melinda Shore
No Mountain Software
PO Box 16271
Two Rivers, AK 99716
US

Phone: +1 907 322 9522
Email: melinda.shore@nomountain.net

Senthil Sivakumar
Cisco Systems
7100-8 Kit Creek Road
Research Triangle Park, NC
US

Email: ssenthil@cisco.com